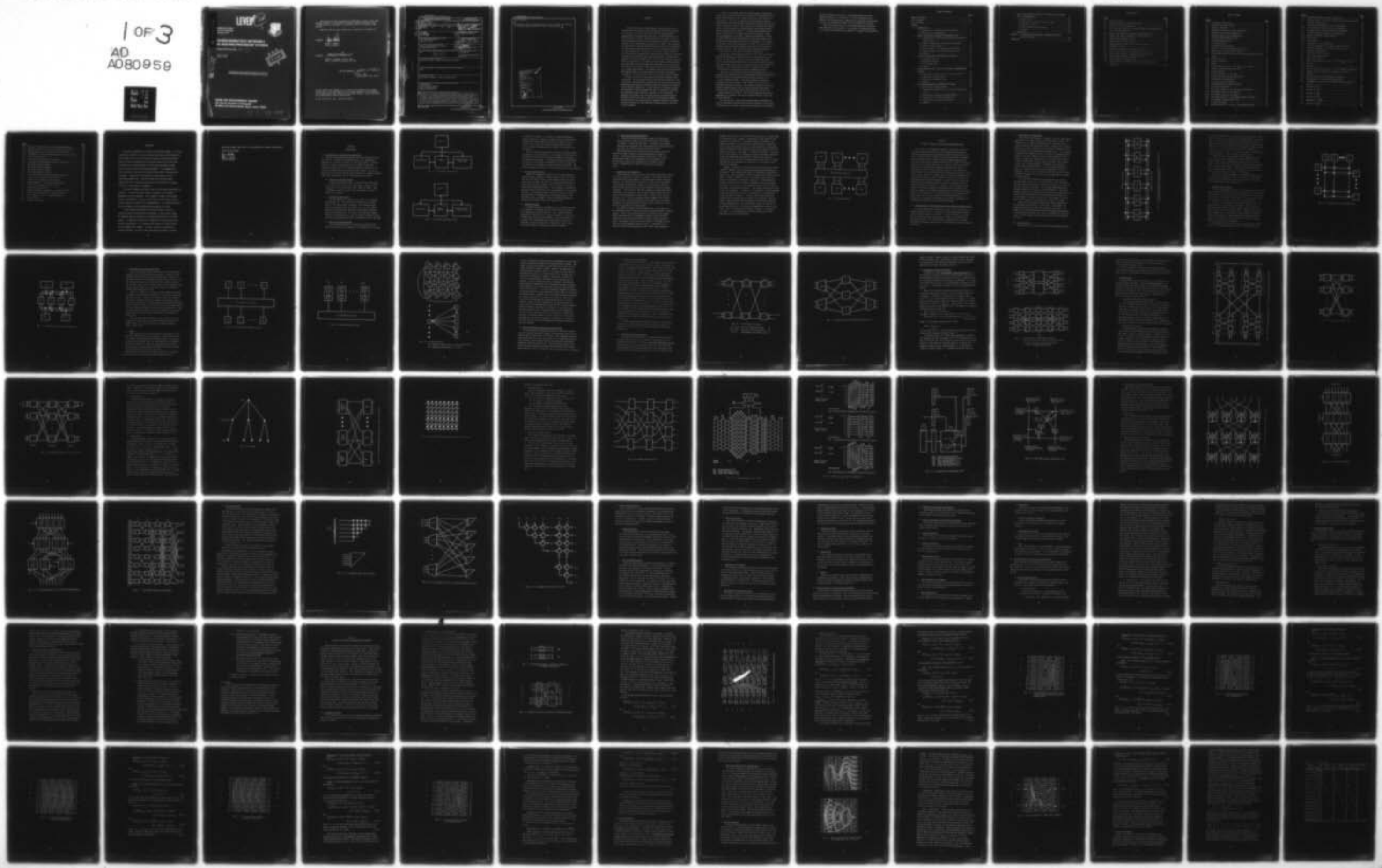
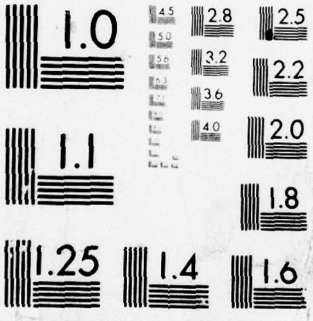
AD-A080 959 WAYNE STATE UNIV DETROIT MICH F/G 9/2
INTERCONNECTION NETWORKS IN MULTIPLE-PROCESSOR SYSTEMS.(U)
DEC 79 T FENG, C WU F30602-76-C-0282
UNCLASSIFIED RADC-TR-79-304 NL

1 OF 3
AD
A080959




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

RADC-TR-79-304
Final Technical Report
December 1979

INTERCONNECTION NETWORKS IN MULTIPLE-PROCESSOR SYSTEMS

Wayne State University 371 550

Tse-Yun Feng
Chuan-lin Wu

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-304 has been reviewed and is approved for publication.

APPROVED:

JAMES L. PREVITE
Project Engineer

APPROVED: Wendall C Bauman

WENDALL C. BAUMAN, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISCA), Griffiss AFB, NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

19 **REPORT DOCUMENTATION PAGE**

READ INSTRUCTIONS BEFORE COMPLETING FORM

18 1. ~~REPORT NUMBER~~
RADC-TR-79-304

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

6 4. TITLE (*and Subtitle*)
INTERCONNECTION NETWORKS IN MULTIPLE-PROCESSOR SYSTEMS

9 5. ~~TYPE OF REPORT & PERIOD COVERED~~
Final Technical Report
May [redacted]76 — June[redacted]79

6. ~~PERFORMING ORG. REPORT NUMBER~~
N/A

10 7. AUTHOR(s)
Tse-yun Feng
Chuan-lin Wu

15 8. ~~CONTRACT OR GRANT NUMBER(s)~~
F30602-76-C-0282

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Wayne State University
Detroit MI 48202

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
62702F
16 55971405 17 14

11. CONTROLLING OFFICE NAME AND ADDRESS
Rome Air Development Center (ISCA)
Griffiss AFB NY 13441

11 12. ~~REPORT DATE~~
Dec[redacted]79

13. ~~NUMBER OF PAGES~~
246

14. MONITORING AGENCY NAME & ADDRESS (*if different from Controlling Office*)
Same

12 244

15. SECURITY CLASS. (*of this report*)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

16. DISTRIBUTION STATEMENT (*of this Report*)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)
Same

18. SUPPLEMENTARY NOTES
RADC Project Engineer: James L. Previte (ISCA)

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)
Multiprocessor
Interconnection Networks
Fault Tolerant Systems
Computer Architecture

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

The class of multistage interconnection networks with the configuration is introduced as a reverse-exchange interconnection network which is shown to be a powerful interconnection network for the parallel processing system. A recursive formula is derived to calculate the control pattern of the network for each of four realizable permutation classes. The recursive formulas can provide superior operating speed over the existing routing algorithms. It is proven that all permutations can be realized by the reverse-exchange network
(Cont'd)

over

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

371 550 slt

Item 20 (Cont'd)

in two passes. Both the construction and routing algorithms are provided. Our results compares favorably with those of other networks.

ABSTRACT

For interconnecting a large number of functional units in a multiple-processor system, the multistage interconnection networks are favorably reviewed in comparison with the interconnection organizations of time-shared/common buses, crossbar switches, and multiport memory schemes. Previous works on the design of multistage interconnection networks are generally related to the network topology or the implementable permutation functions. The influence of communication protocols which should, nevertheless, be implemented for the intercommunication function of the interconnection network is usually neglected. In addition, this field is still lacking a set of performance standard and evaluation tools which can be employed to observe the tradeoffs among various parameters. Besides these problems, the fault diagnosis scheme for the interconnection networks, which is important for a reliable or fault tolerant systems, has not been developed, and the multiple-pass realization of an interconnection network and related routing algorithms have only been discussed for a single stage network. The problem concerning the cost-effective LSI implementation of the interconnection networks also remains unanswered. This study offers relevant solutions to these problems.

We begin our study by surveying the multiple-processor interconnections. In this survey we first discuss the limitations of the conventional interconnection organizations and then review particular multistage interconnection networks which were proposed from significantly different viewpoints. A wide variety of switching concepts and parameters, which a designer may have to encounter in planning and designing an interconnection network, is also summarized. In addition, we provide a set of characteristics of interconnection networks, which can be used to specify the performance standard. We also describe the requirements of hardware facilities and communication protocols for cost-effective implementations of interconnection networks.

A class of multistage interconnection networks is defined by introducing a baseline network and a condition of topological equivalence, and proving that the condition holds for every network so far proposed. The class of topologically equivalent multistage interconnection networks includes the regular SW banyan network with S=F=2, the indirect binary n-cube network, the modified data manipulator, the flip network, the omega network, the baseline network, and the reverse baseline network. A logical name representation scheme is developed to configure this class of networks. Using this configuration, we propose a complete and homogeneous routing procedure which includes capabilities of resolving conflicts and allowing connections between all pairs of terminals. The routing can be done in both ways (from side 1 to side 2, or vice versa) in contrast to the previously proposed one which can only be done in a specific direction. Our routing procedure implicitly provides a routing protocol for the packet switching communication.

We present a fault diagnosis scheme for the class of multistage networks by proposing a general fault model and generating a test set for the fault model. Specific steps for diagnosing single faults and detecting multiple faults in the interconnection network such as the flip network and the indirect binary n-cube network are developed. This study provides specific information of fault characteristics for designing an easily diagnosable network.

The class of multistage interconnection networks with the configuration is introduced as a reverse-exchange interconnection network which is shown to be a powerful interconnection network for the parallel processing system. A recursive formula is derived to calculate the control pattern of the network for each of four realizable permutation classes. The recursive formulas can provide superior operating speed over the existing routing algorithms. It is proven that all permutations can be realized by the reverse-exchange network in two passes. Both the construction and routing algorithms are provided. Our result compares favorably with those of other networks.

We then describe a logic partitioning scheme to implement the class of multistage interconnection networks optimally in the sense of using LSI circuit chips of one type and resulting in the maximum switching element-to-pin ratio. The scheme extends the switch design from size 2 × 2 to size $2^{\alpha} \times 2^{\alpha}$, thus facilitating cost-effective LSI implementation, improving the reliability of switching elements and formulating another level of problems for research.

Some future needs and a set of possible extensions related to this study are also discussed. A case study for implementing part of the design philosophy we just exposed is provided in the Appendix.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS i

LIST OF TABLES iii

LIST OF FIGURES iv

CHAPTER

1 INTRODUCTION 1

1.1 Definition of a Multiple-Processor System 1

1.2 Objective of Investigation 4

2 SURVEY OF MULTIPLE-PROCESSOR INTERCOMMUNICATIONS 7

2.1 Classification of Multiple-Processor Interconnection Organizations 7

2.2 Review of Multistage Interconnection Networks 17

2.3 General Design Criteria 46

2.4 Characteristics of Interconnection Networks 48

2.5 Network Hardware and Software Design Issues 50

2.6 Summary 56

3 A CLASS OF MULTISTAGE INTERCONNECTION NETWORKS 57

3.1 Isomorphic Topology 57

3.2 Routing Techniques 74

3.3 Full Communication 82

3.4 Summary 87

4 FAULT-DIAGNOSIS FOR A CLASS OF MULTISTAGE INTERCONNECTION NETWORKS 89

4.1 Fault Model and Test Set of a Switching Element 89

4.2 Diagnosis of Single Faults 93

4.3 Detection of Multiple Faults 132

4.4 Summary 133

5 THE REVERSE-EXCHANGE INTERCONNECTION NETWORK 136

5.1 The Reverse-Exchange Network 137

5.2 Permutations Realizable by the Reverse-Exchange Network 141

5.3 Controlling the Reverse-Exchange Network 145

5.4 Realization of Arbitrary Permutations 157

5.5 Applications on Parallel Processing 168

5.6 Summary 170

6 LOGIC PARTITIONING OF MULTISTAGE INTERCONNECTION NETWORKS FOR LSI IMPLEMENTATION 172

6.1 Partitioning 173

6.2 Minimizing the Number of Modular Types 179

6.3 Analysis on Pins 181

6.4 Interconnecting Circuit Chips 187

6.5 Summary 193

7 CONCLUSION 194

APPENDIX - A MICROPROCESSOR-CONTROLLED ASYNCHRONOUS CIRCUIT SWITCHING NETWORK 200

REFERENCES 224

LIST OF TABLES

Table		Page
3.1	A Conflict Table	81
3.2	A Reduced Table of Conflict Resolution	83
3.3	Result of Conflict Resolution	84
4.1	Set of the 16 States and the Related Symbolic Representation of a 2×2 Switching Element	91
4.2	Faults, Test Inputs and Outputs in Valid State S_{10}	92
4.3	Faults, Test Inputs and Outputs in Valid State S_5	94
4.4	Faulty Output Pattern in Case 1	106
4.5	Faulty Output Pattern in Case 2	112
4.6	Classification of the Functional State in Case 2	112
4.7	Examples for Case 2	114
4.8	Faulty Output Pattern in Subcase A of Case 2	116
4.9	Faulty Output pattern in Subcases B and C of Case 2	119
4.10	Faulty Output Pattern in Subcases D and E of Case 2	125
4.11	Faulty Output pattern in Subcase F of Case 2	129
4.12	Characteristics of Single Faults	134
A.1	Status Table of an 8×8 Baseline Network for Asynchronous Operation	219

LIST OF FIGURES

Figure Page

1.1 Von Naumann organization 2

1.2 A parallel processor system 2

1.3 Network - PMS model 6

2.1 Time-shared/common bus system organization 9

2.2 Crossbar switch system organization 11

2.3 Multiport memory system organization 12

2.4 Processor-Network-Memory system 14

2.5 Processor-Network system 15

2.6 A connection grid 16

2.7 A three-stage Clos network 19

2.8 Cantor's strictly nonblocking construction 20

2.9 Construction of Benes binary networks 22

2.10 A four-stage network which is found in many telephone central offices 24

2.11 Series connection 25

2.12 Parallel connection 26

2.13 Banyan 28

2.14 L-level banyan structure with fanout F and spread S 29

2.15 3-level CC banyan network with S=F=2 30

2.16 An omega network 31

2.17 A data manipulator for 8 items 33

2.18 Control group for the data manipulator 34

2.19 A versatile line manipulator 35

2.20 The logic circuit of BLMC cell 36

2.21 An expanded shuffle-exchange network 38

2.22 An 8-item flip network 39

2.23 An 8-item network for flip and shift permutations 40

2.24 The indirect binary 4-cube network 41

2.25 A triangular array and its symbol 43

2.26 Clos construction for a one-sided triangular network 44

2.27 A triangular interconnection array 45

3.1 A switching element 59

3.2 Recursive process to construct the baseline network 59

Figure Page

3.3 A baseline network with name representation 61

3.4 A banyan (S=F=2), or indirect binary n-cube network structure with new configuration 64

3.5 A modified data manipulator with configuration 66

3.6 A flip network structure with new configuration 68

3.7 An omega network structure with new configuration 70

3.8 A reverse baseline network with configuration 72

3.9 Binary tree coding of omega network 76

3.10 Path routing 78

3.11 A full switch 86

3.12 An example of full communication 86

4.1 Test set and response for a basic composite network 96

4.2 Fault-free response of a network tc the test set 97

4.3 Examples of observations 99

4.4 Locating the link stuck fault 103

4.5 Case 1 (one faulty output) 108

4.6 Experiments in Case 1 109

4.7 Test to differentiate $\phi\phi$ and -- for the example in Case 1 ... 111

4.8 Subcase A 117

4.9 Subcase B 121

4.10 Test to differentiate $\phi\phi$ and -- for the example in Subcase B 122

4.11 Summary of test procedures for Subcases D, E and F 124

4.12 Subcase D 127

4.13 Blocks of faulty location pattern of Subcase F 131

5.1 Configuration of a reverse-exchange network 138

5.2 A permutation realized by the reverse-exchange network of size 2^3 140

5.3 Setting for $F_3^{(3)}$ 148

5.4 Setting for $\tilde{F}_4^{(3)}$ 148

5.5 Setting for $C_{5,7}^{(4)}$ 151

5.6 Setting for $\tilde{C}_{5,7}^{(4)}$ 153

5.7 Permutation of $S_{2,3}^{(4)}$ 155

5.8 Setting for $S_{2,3}^{(4)}$ 158

Figure Page

5.9 Two-pass construction for a reverse-exchange network 159

5.10 Equivalent construction of the two-pass construction 160

5.11 Another structure of the equivalent construction 162

5.12 Confinement of some switching elements in the second pass ... 163

5.13 A Benes binary network 164

5.14 A setting of the Benes binary network 167

5.15 A setting of the equivalent two-pass construction 169

6.1 The first partition example 174

6.2 The second partition example 175

6.3 The third partition example 176

6.4 The fourth partition example 177

6.5 Seven valid states in the asynchronous operation 183

6.6 A module for the asynchronous operation 185

6.7 Example for the partial implementation 190

6.8 Name assignment in the partial implementation 192

A.1 A model of a multiple-processor system 204

A.2 The network configuration 205

A.3 Explicit hardware requirement in a connection path 207

A.4 Functional block diagram of interface processor 209

A.5 Block diagram of an N × N switching element 210

A.6 Global routing 214

A.7 Example of N/2 paths 218

EVALUATION

Large Scale Integration is causing revolutionary changes. It is now economically feasible to construct processing systems by interconnecting large numbers of low cost off-the-shelf processors and memory modules. Physical limits and economics are providing strong bias for utilizing a multiplicity of these modules to attain processing power through parallelism and reliability through redundancy. To accommodate this trend, particular focus must be placed on multi-module interconnection strategies. Conventional interconnection organizations such as time-shared/common buses, crossbar switches and multiport memory schemes have their limitations and are not quite suitable for systems involving a large number of components.

This report addresses the whole class of multi-stage interconnection networks. A formal addressing scheme is developed to facilitate a homogeneous routing procedure. Fault diagnosis schemes are developed. Finally, consideration is given to partitioning a multi-stage interconnection network for practical LSI implementation. This partitioning minimizes the number of chips and maximizes the gate to pin ratio.

The multi-stage interconnection network provides a basis for exploiting low-cost mainstream LSI technology to provide wide ranges of computing power by economically configuring collections of standard modules. In addition to providing a framework for fault tolerance and modular system growth, it is expected that software for these systems will be somewhat less complex. As such, TPO thrusts 3D and 5A are directly affected. The multi-stage interconnection network is being emulated on RADC's QM-1 and will be available for further investigation relative to C^3 goals.

JAMES L. PREVITE
Project Engineer

CHAPTER 1

INTRODUCTION

1.1 Definition of a Multiple-Processor System

Various multiple-processor systems have been proposed and constructed [1]. The specific examples include four organizations (associative, parallel, pipeline, and multi-processors). Although these four classes of organizations are a subset of multiple-processor systems, there is no existing definition which can conclusively characterize every attribute in the subset. In the following we first discuss some existing definitions of computer systems, then offer our definition of the multiple-processor system:

A. Instruction and Data Streams

Flynn [2] proposed in 1966 a classification scheme based on the instruction streams and data streams. There are four categories in the scheme (SISD, SIMD, MISD, and MIMD). Many contemporary computer systems can be classified in terms of these four categories.

B. Von Neumann Architecture

The vast majority of computers are based on von Neumann's architecture [3]. The architecture includes four principal units -- the control unit, the memory, the arithmetic-logic unit, and the input/output unit (Fig. 1.1). It is a single processor system with the capability of sequential execution, known as an SISD machine. As a result of technology changes, the central processing unit in the von Neumann architecture has evolved to multiple-execution-unit organization such as CDC 6600.

C. Parallel Processing Systems

The multiple-execution-unit organization provides some concurrent activities and sometimes is considered to be one form

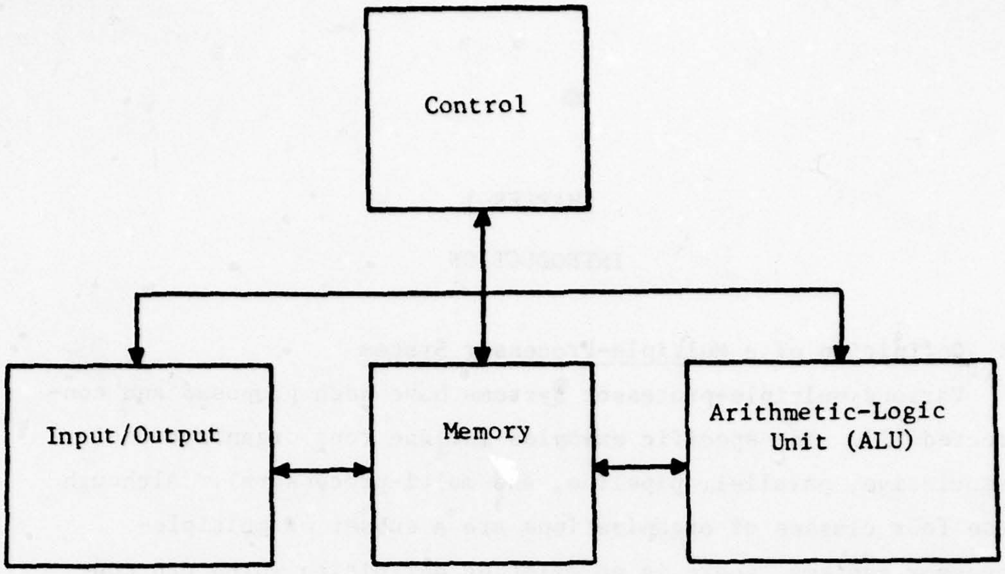


Fig. 1.1 Von Neumann organization

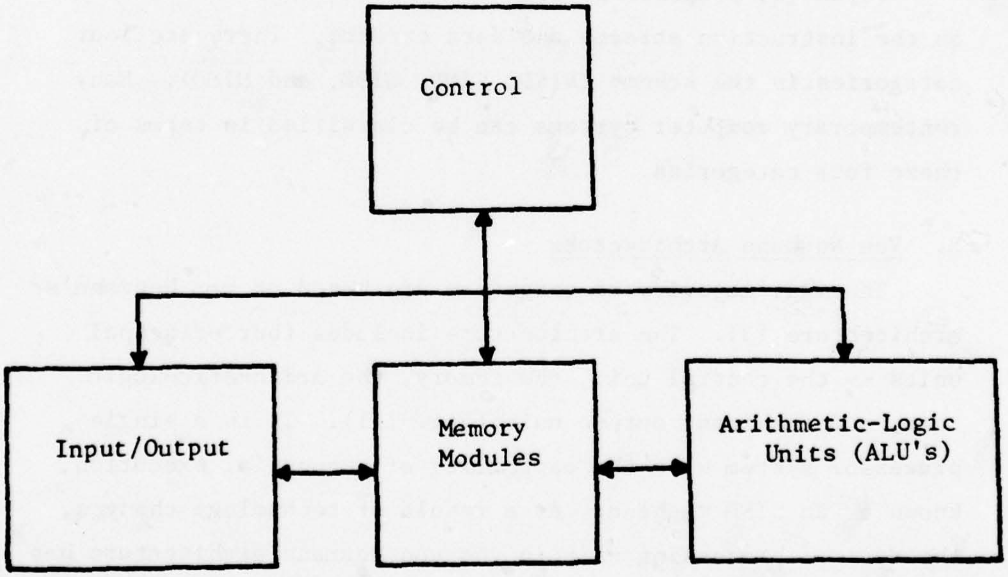


Fig. 1.2 A parallel processor system

of parallel processing. In general, a parallel processing system consists of a number of arithmetic-logic units (known as processing elements) and memory modules as shown in Fig. 1.2. In most cases, a parallel processor system requires interprocessor/memory communications to improve its capability and performance.

Feng [4] classified associative/parallel processing systems according to the word length, i.e., the number of bits which are processed in parallel in a word and the number of words which are processed in parallel. A computer structure is represented by a point in a plan where the abscissa is the word length and the ordinate is the number of words processed in parallel. The parallel processors include two types: homogeneous and heterogeneous.

D. Multiprocessor Systems

Enslow [5] suggested a definition for the multiprocessor system as being a subclass of MIMD systems in which the processors have the common access to their primary memory as well as input-output channels, and there is a single operating system controlling the entire complex. Baer [6] used two features to differentiate MIMD architecture -- the coupling or switching of processor units and memories, and the homogeneity of the processing units. He discussed tightly coupled and loosely coupled multiprocessors of homogeneous and heterogeneous types. The CDC 6600 is thus an example of tightly coupled heterogeneous multiprocessor systems under this classification.

E. Computer Networks

A computer network is considered to be any interconnection of an assembly of computer systems and/or terminals together with communications facilities [7]. Such a network permits geographical distribution of computer operation, parallel processing, and various resource sharing. The basic attributes of a network that distinguishes its architecture includes its topology or overall organization, composition, size, channel type and utilization strategy, and control mechanism.

F. Distributed Processing Systems

Enslow [8] defined an allowable region for distributed data processing systems in the decentralization space formed by hardware, control and data base. In general, the multiple processors and computers with cooperating control and partitioned data base are distributed data processing systems.

The characteristics and classifications of machines described above may be used to define the multiple-processor systems as follows: a multiple-processor system contains two or more functional modules which can be homogeneous or heterogeneous; and the functional modules are interconnected to achieve various levels of capabilities (at least as those stated previously [2-8]).

1.2 Objective of Investigation

Recent advances in LSI technology have caused significant changes. It is now economically feasible to construct a processing system by interconnecting a large number of off-the-shelf processor and memory modules. The architectural trend [9-11] is thus to use a plurality of processors interconnected together to gain increased operating power through parallelism and to improve system reliability through redundancy. On the basis of the past progress in integrated-circuit technology, it is projected that the most complex computer system of today can be fabricated on a small number of chips within the next few years [12]. This LSI technology project suggests that complex dynamic modules of processor-memory-switch (PMS) group can be made available for constructing the multiple-processor system in the same way as the architectural trend predicts.

The number of dynamic modules (homogeneous or heterogeneous) in the multiple-processor system would keep increasing because of several reasons. First, the processing speed in the future can be significantly increased only by increasing the degree of the concurrent processing as the switching speeds of computer devices approach a limit. Furthermore, there are certain classes of problems, such as large data base management systems, weather computations, etc., which are beyond the capabilities of the current large computers.

Thirdly, the low cost of LSI modules allows the use of a large number of functional units. However, coordinating a large number of PMS groups into an efficient functional system is a difficult yet important problem. The conventional interconnection organizations such as time-shared/common busses, crossbar switches, and multiport memory schemes have their limitations and are not quite suitable for systems involving a large number of components (modules). The objective of this study is to investigate the interconnection problems by using some multistage interconnection networks upon which the multiple-processor system can be modelled as shown in Fig. 1.3 where the PMS box could represent any combination of processors, memory modules and switches.

Chapter 2 reviews the multiple-processor interconnection organizations in general, and the multistage interconnection networks in particular. Some communication issues of multistage interconnection networks are also discussed in terms of hardware and software requirements. Chapter 3 defines a class of multistage interconnection networks by showing the isomorphic topology. A formal addressing scheme is developed to facilitate a homogeneous routing procedure. In Chapter 4, we develop a fault-diagnosis scheme for the class of multistage interconnection networks. Both single faults and multiple faults are considered. In Chapter 5, we introduce a reverse-exchange interconnection network which is a consequence of the addressing scheme developed in Chapter 3. Both the realizable permutation classes and the related control patterns are developed. Chapter 6 shows a logic partitioning of multistage interconnection networks for LSI implementation. The partitioning is made in the sense of minimizing the number of chip types and maximizing the gate-to-pin ratio. After providing the conclusion in Chapter 7, we present a case study -- a microprocessor-controlled asynchronous circuit switching network in the Appendix.

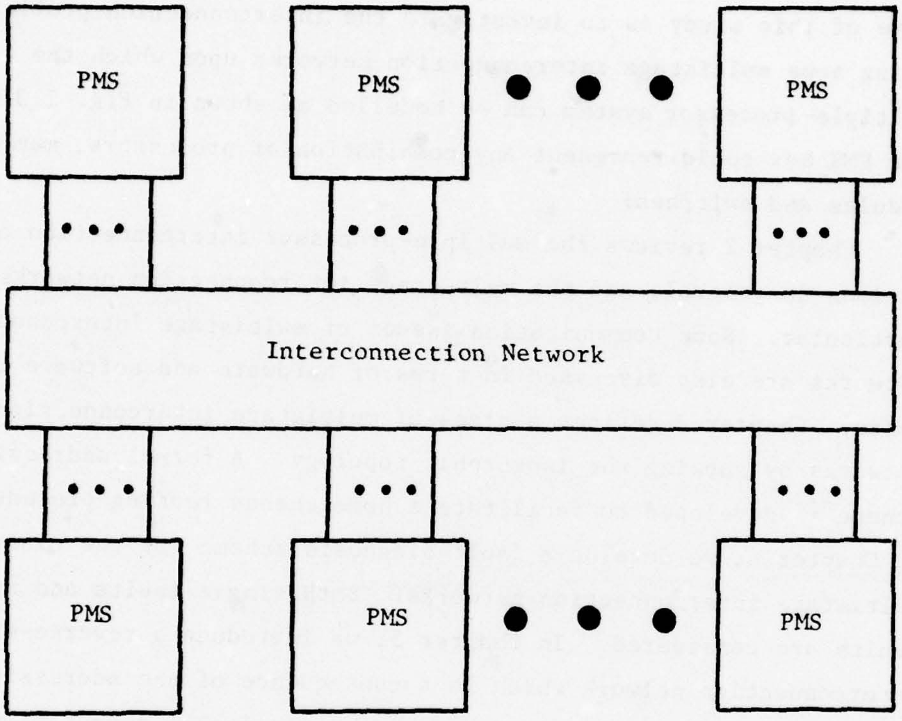


Fig. 1.3 Network-PMS model

CHAPTER 2

A SURVEY OF MULTIPLE-PROCESSOR INTERCOMMUNICATIONS

As a result of increasing the number of functional modules in the multiple-processor system, the intercommunications among the functional modules become increasingly complex and inevitably necessary. Interconnection networks have been investigated to implement the intercommunications. However, general design guidelines and practical issues such as communication protocols, dynamic reconfigurations, etc. are usually neglected. In addition, this field is still lacking a set of performance standards and evaluation tools for designing an efficient interconnection network. This chapter provides a survey on the intercommunication issues with emphasis on these future needs. Section 2.1 reviews the interconnection organizations of multiple-processor systems to emphasize again the importance of multistage interconnection networks. Section 2.2 surveys particular multistage interconnection networks which were proposed from significantly different viewpoints. In Section 2.3, we catalogue a wide variety of switching concepts and parameters which a designer may have to encounter in planning and designing an interconnection network. Section 2.4 provides a set of characteristics of interconnection networks, which can be used to specify the performance standard. In Section 2.5, we describe some hardware and software requirements for implementing functions of interconnection networks.

2.1 Classification of Multiple-Processor Interconnection Organizations

The intercommunication subsystem is a key to the classification of computer systems [9,13,14]. The scope of intercommunication schemes can be viewed in several levels [15]. In this section we will emphasize the interconnection organizations in order to characterize the system architecture. The interconnection organizations of present day multiple-processor systems can be classified into several categories as follows:

A. Time-Shared or Common Buses:

There are several degrees of complexity in this organization depending on the number and the functional usage of buses. The simplest one is a common communication bus connecting all processors, memories and input-output units. The bus can be totally passive and transfer operations are controlled completely by the bus interfaces of the sending and receiving units. It is possible to simplify the transfer process by the use of a centralized bus arbiter. The cost required to add or remove functional units to the bus is quite low. Usually all that is required is, within a limit, to physically attach or detach the unit. The location to add a unit is also flexible. The single bus introduces a critical system component which can cause a system failure as a result of a malfunction in any circuit component of the bus subsystem. There is also a serious bottleneck on overall system performance because only one path can be established at any time for data transfers.

To obtain more reliability and parallelism, one can use multiple buses, either uni- or bi-directional. The intercommunication functions may be partitioned and a separate bus is used for each functional partition. On the other hand multiple and redundant buses may be used for system reliability. The use of multiple buses also allows multiple simultaneous transfers (Fig. 6.2). However, any benefit derived from the multiple-bus schemes is at the expense of complex bus controls. Some degrees of control logic such as arbitor or multiplexor would have to be added.

Some examples of systems employing the simple time-shared bus technique include PDP-11, Lockheed Sue, and CDC 6600 (for transfers between main memory and peripheral processors). An example for functionally partitioned and redundant multiple buses is proposed by GTE Sylvania Inc. [16] for communication applications. Another example for multiple buses is the Plessy System 250 [13].

B. Crossbar Switch:

The crossbar switch provides nonblocking simultaneous memory

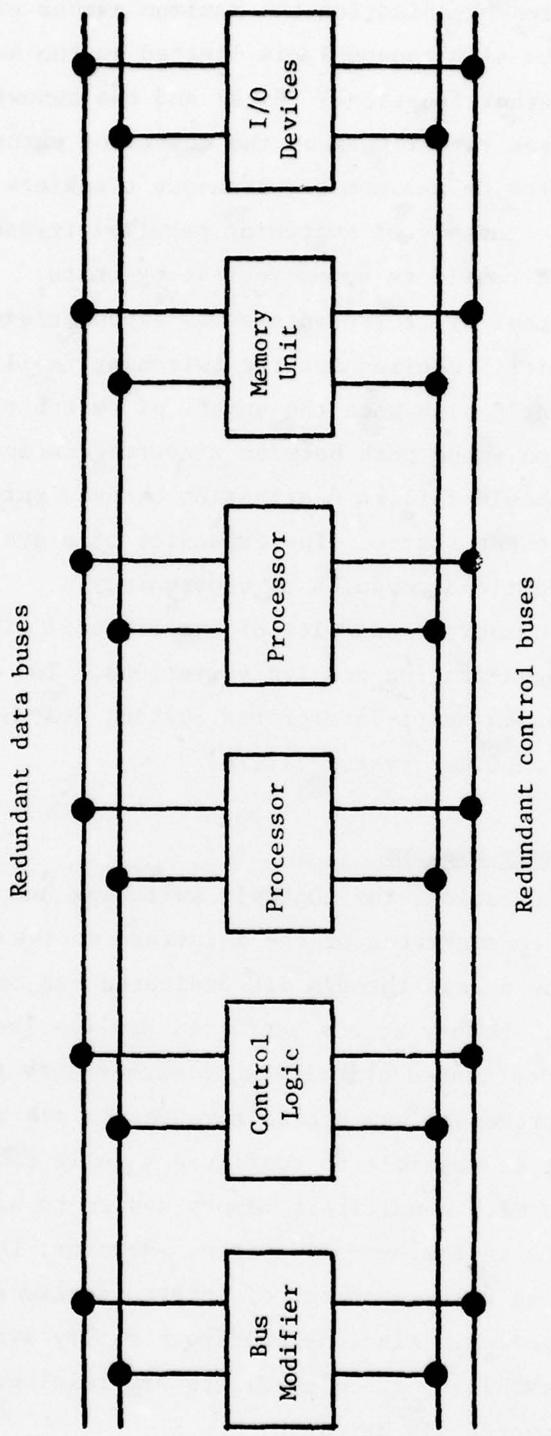


Fig. 2.1 Time-shared/common bus system organization (functionally partitioned, multiple two-way buses).

accesses and communications among other functional units. With this interconnection organization the maximum number of transfers that can take place simultaneously is limited by the number of memory units (or other functional units) and the bandwidth-speed product of the buses rather than by the number of paths available (Fig. 2.2). To provide maximum simultaneous transfers, each crosspoint must be capable of switching parallel transmission and resolving possible conflicts among requesting units.

Since the number of crosspoints grows exponentially, the cost of the circuitry required for the switching facilities becomes significantly high when the number of switch ports is large. There is only one path between a source-destination pair. If a crosspoint should fail, a destination becomes unreachable from the correspondent source. The expansion of a system can be done by adding additional modules of crosspoints.

There are a number of examples of systems utilizing crossbar interconnection organization and its variations. The examples include the Burroughs Multi-Interpreter System, RCA 215 system, and Carnegie-Mellon C.mmp system [17,18].

C. Multiport Memory Systems

In this organization, the control, switching and priority arbitration logic is concentrated at the interface to the memory units. Each processor has access through its dedicated bus to all memory units (Fig. 2.3). Memory access conflicts are resolved by assigning permanently designated priorities to each memory port.

Since each processor has access through its own bus to all memory modules it is possible to configure a fully connected crossbar topology with a multiport memory system to have a very high transfer rate in the overall system. However, the multiport memory systems have a large number of interconnections between processor and memories. Also the multiport memory system has limits on its flexibility since conflicts are resolved through priorities implemented via hardware.

The multiport memory organizations are often found in large systems. The examples include the Univac 1108 system and the

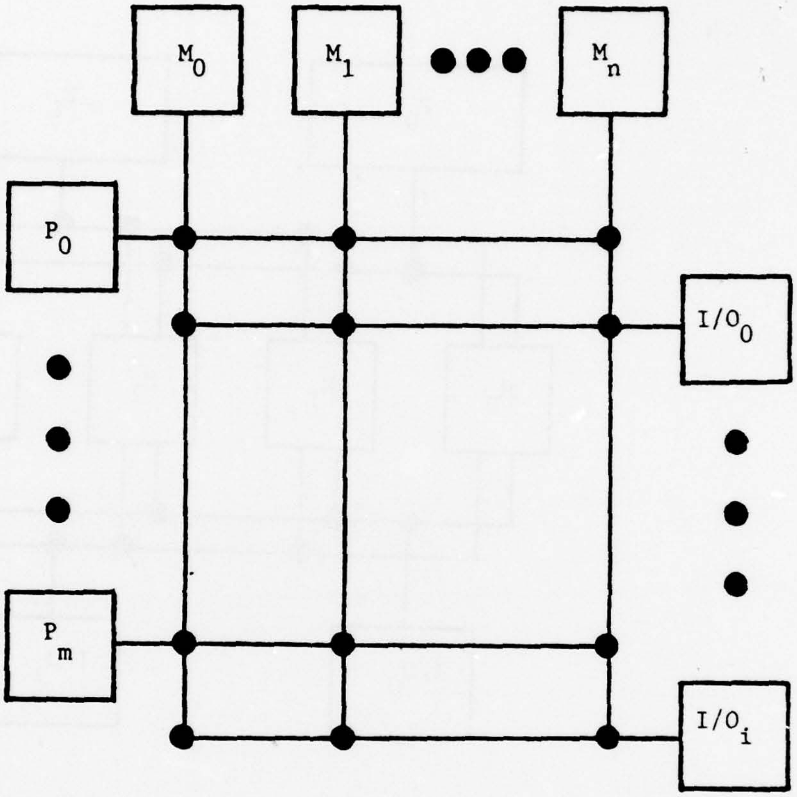


Fig. 2.2 Crossbar switch system organization.

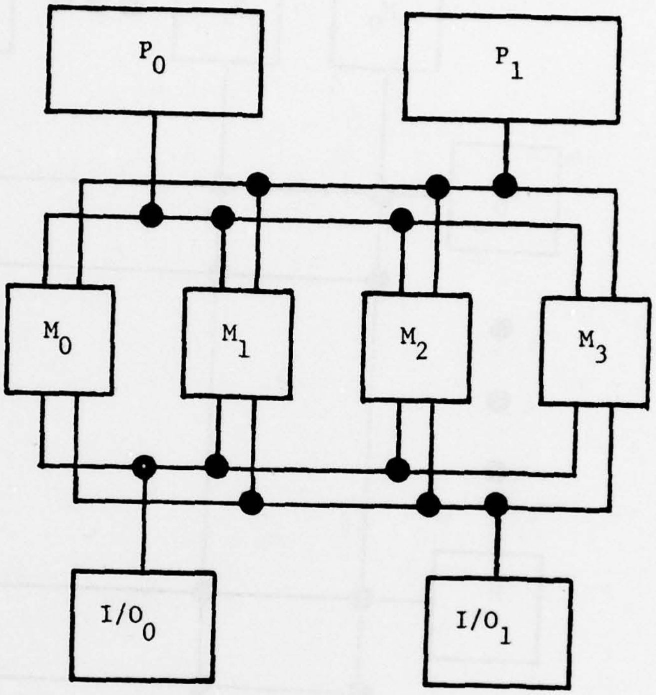


Fig. 2.3 Multiport memory system organization.

D. Multistage Interconnection Networks

Many of the large computer systems that are being used and designed today have interconnection networks. In most of these systems, the interconnection networks constitute the heart of the systems which can be modelled as shown in Fig. 2.4 or Fig. 2.5. The essential elements of an interconnection network include a set of input lines, a set of output lines, a set of control lines and the related connective logic which consists of switching elements and communication links.

There are a number of variations of interconnection networks depending on the functional requirements, the control scheme and many other factors. Usually the connective logic is organized into several stages of switching elements and connection links to achieve full connections. The multistage interconnection networks allow processor-to-memory and processor-to-processor communications in a more general way than the other three organizations do. When a system consists of numerous functional modules the multistage interconnection network becomes the dominant interconnection organization.

The interconnection organization used in ILLIAC IV can be considered as a special case of the multistage interconnection network (Fig. 2.6). The flip network in STARAN represents another example [19].

E. Others

Some systems have mixed interconnection organization. The Pluribus system [20,21] is an example [6]. This system is used as a modular switching node for the ARPA network. It consists of seven processor buses with two processors and two 4K memories attached to each other. There are also memory and I/O buses. The seven dual processors share two banks of two memory modules in an organization of multiport structures.

Reviewing the evolution of the computer systems, one can find that for the past three decades tremendous progress in device,

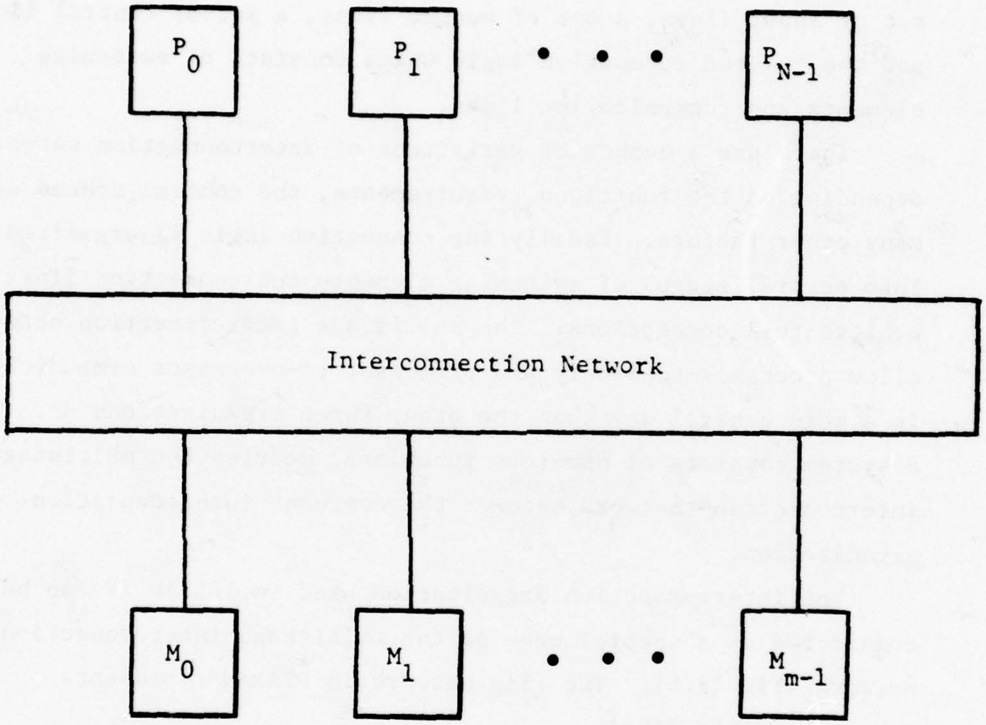


Fig. 2.4 Processor-Network-Memory system.

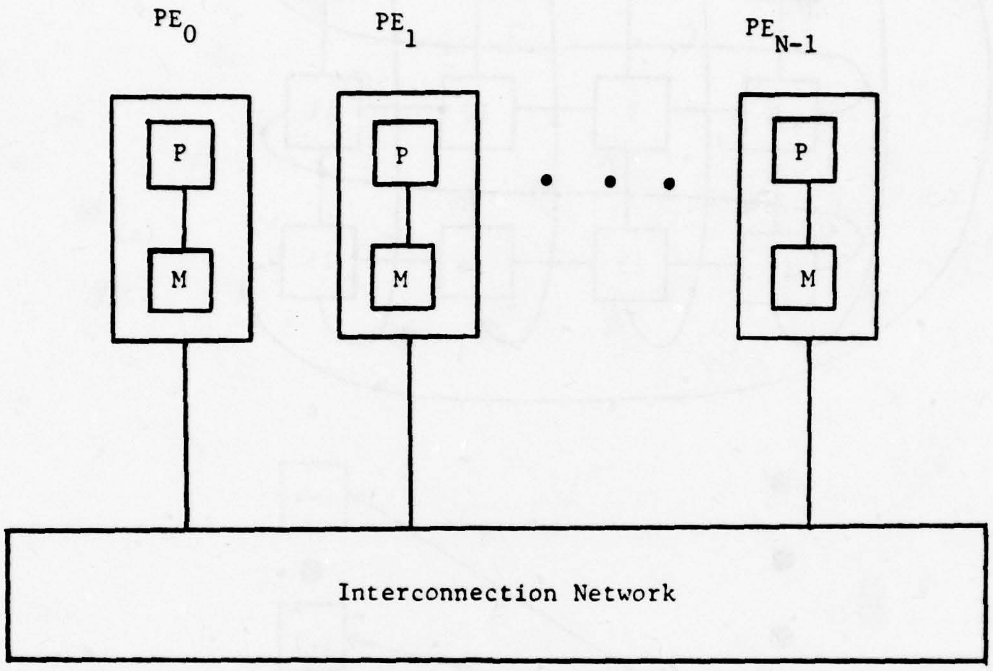


Fig. 2.5 Processor-Network system.

30

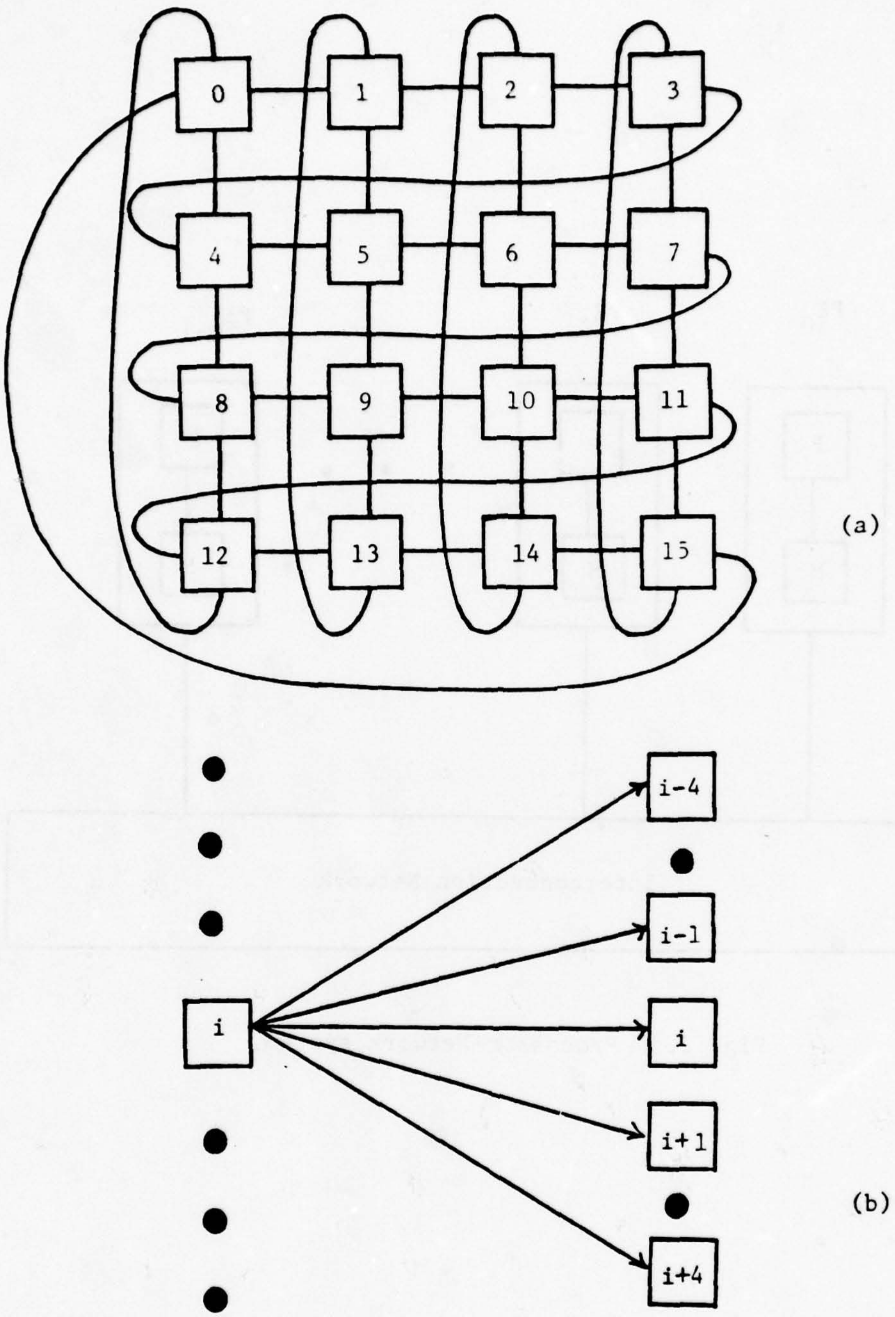


Fig. 2.6 A connection grid.

(a) Connection grid similar to that used in ILL IV.

(b) Equivalent connection, 0 ≤ i ≤ 15.

circuit technology and miniaturization techniques have been used to construct high speed sequential processing computer systems. The number of functional modules is fixed and limited, and the functional modules are tightly coupled. However, as improvement in switching speed reaches a limit, it is obvious that any further significant increase in processing speed can be obtained only by concurrent processing of a number of data sets. To this end, various multiple-processor systems have been proposed and constructed. We are reaching the point to develop a new efficient interconnection organization for such multiple-processor systems. Recent advances in LSI technology also facilitate this new architecture trend [9,10]. It is now economically feasible to construct a processing system by interconnecting a large number of processors and memory modules. However, when the number of functional modules in a system increases to a certain level, say the order of 100, the choice of interconnection organizations becomes a critical problem. People are even considering to obtain a multiple-processor system by interconnecting as many as 10^5 functional modules. System performance and practical feasibility of such multiple-processor systems would be terribly limited if the conventional interconnection organization such as time shared or common bus, crossbar, or multiport memory is used. Thus one of the exciting challenges in the field of computer architecture is to design an efficient and practical intercommunication subsystem of multiple-processor systems.

2.2 Review of Multistage Interconnection Networks

As interconnection organization becomes an important research topic for multiple-processor systems, the multistage interconnection networks should receive special attention. The multistage interconnection networks are used in many areas such as telephone switching, data alignment between memory modules and processors, permutation generators, and data sorting, etc. This section provides a brief review on these multistage interconnection networks (with N inputs and N outputs) which are classified into the following four categories:

A. Strictly Nonblocking Network

A network which can connect any idle input to any idle output regardless of what other connections are current in progress is called a strictly nonblocking network. There exist at least three construction methods. The rectangular switch with N x N crosspoints is a strictly nonblocking one. In 1953, Clos gave an explicit construction [22]. A three-stage Clos nonblocking network constructed from a number of smaller crosspoint switches (matrix boxes) is shown in Fig. 2.7 with $m \geq 2n-1$. Clos has shown that the network has less than N^2 crosspoints for $N \geq 24$ and in general it requires asymptotically $cN \exp[2 \sqrt{\log_2 N}]$ crosspoints. The type of networks shown in Fig. 2.7 can be generalized to the $2k-1$ stage case by iteratively applying the construction rule to matrix boxes in the center stage $k-1$ times. The third method given by Cantor [23,24] is based on a three-stage network somewhat related to the Clos three-stage networks. The construction is shown in Fig. 2.8 in which M is a nonblocking network and L' is the mirror image of L. L is not nonblocking but it is designed in such a way that regardless of the state it is in, if there is an idle input there are available paths to connect it to more than b/2 of its outputs (Fig. 2.8). It requires asymptotically $2N(\log_2 N)^2$ crosspoints. For $N > 100{,}000$ the Clos network is not as good as the Cantor network in terms of crosspoint count.

Strictly nonblocking networks have found very limited applications since some alternative networks which are simpler to manufacture and to control can be built with very low blocking probability.

B. Wide Sense Nonblocking Network

A network which can handle all possible connections without blocking but can do so only if specific routing rules are used to make its connections is called a wide sense nonblocking network. As pointed out in [25], Clos networks with $m = [3n/2]$ are wide sense nonblocking and such networks have fewer crosspoints than the strictly nonblocking Clos network ($m = 2n - 1$) with the same

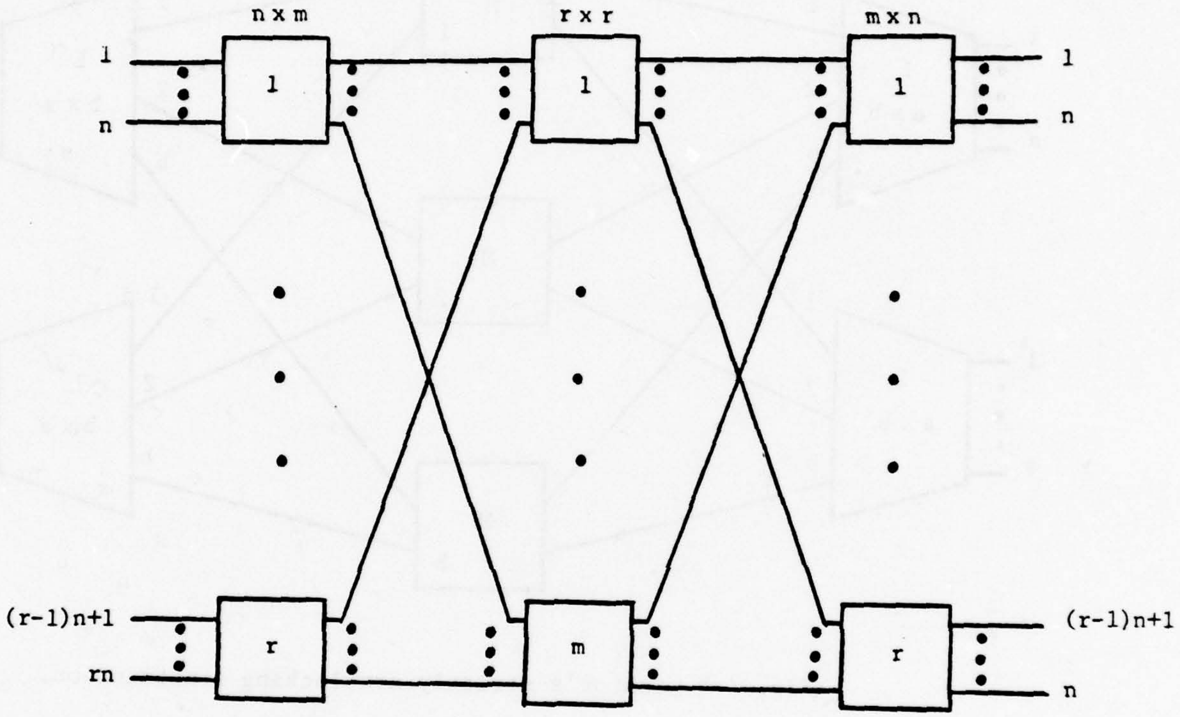


Fig. 2.7 A three-stage Clos network

$$\left(\begin{array}{ll} m \geq 2n-1 & \text{strictly nonblocking network} \\ m \geq \frac{3}{2}n & \text{wide sense nonblocking network} \\ m \geq n & \text{rearrangeable nonblocking network} \end{array}\right).$$

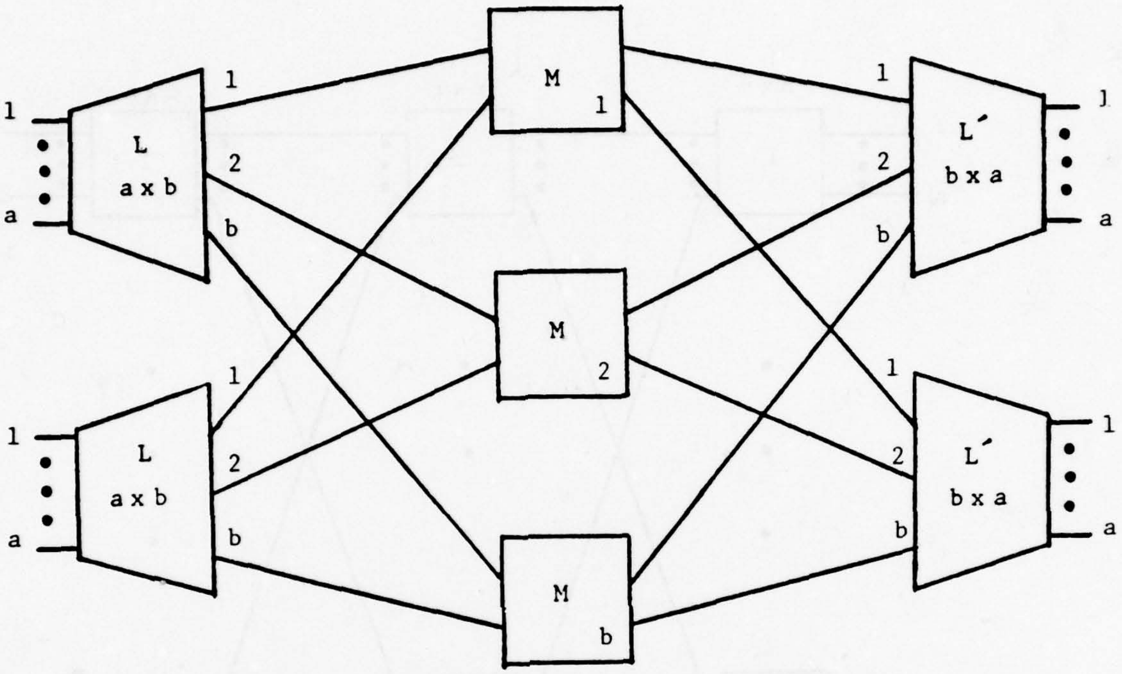


Fig. 2.8 Cantor's strictly nonblocking construction.

number of inputs. However, there is no proof whether this holds for more general cases. Furthermore, there may be no advantage in requiring only the wide sense nonblocking condition rather than the strict sense nonblocking condition.

C. Rearrangeable Nonblocking Network

A network is called rearrangeable nonblocking network if it can perform all possible connections between inputs and outputs by rearranging its existing connections so that a connection path for a new input-output pair can always be established. The class of networks shown in Fig. 2.7 is rearrangeably nonblocking if and only if $m \geq n$. The above statement is called the Slepian-Duguid theorem.

Benes [26] has given the general construction scheme for a multistage rearrangeable network to have the minimal number of crosspoints. If N can be factored into its k prime factors, $N = 2^{\alpha_1}3^{\alpha_2}\ldots p_k^{\alpha_k}$, Benes algorithm leads to a $(2\sum_{j=1}^{k} \alpha_j - 1)$ - stage network if the largest prime factor exceeds three or if it equals three and N is odd, or a $(2\sum_{j=1}^{k} \alpha_j - 3)$ - stage network if the largest prime factor equals three and N is even. The number of crosspoints required is equal to C(N) and

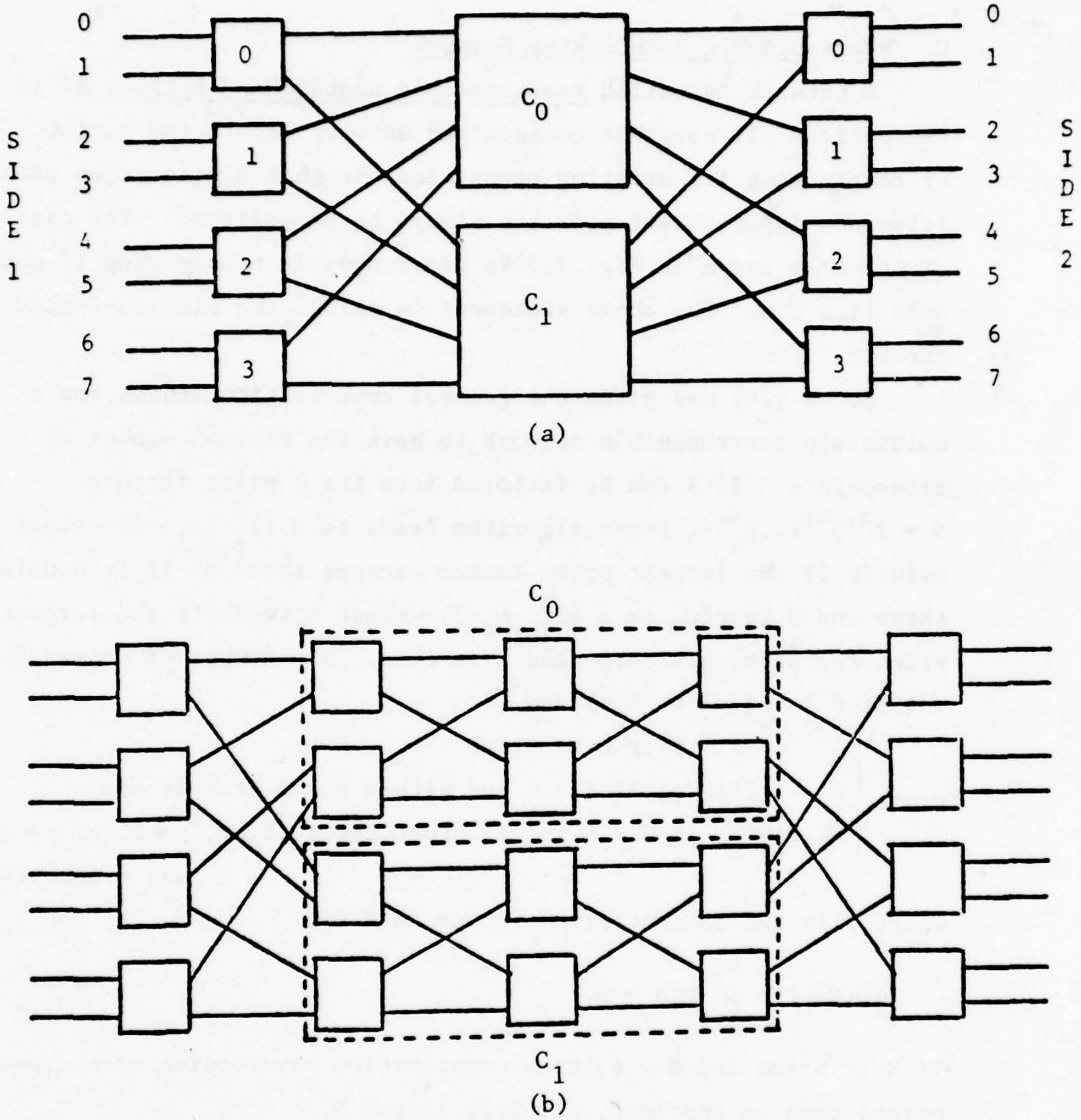
$$C(N) = \begin{cases} N^2 & \text{if } N \leq 6 \text{ or } N \text{ is prime} \\ N(p + 2D(N/p) & \text{if } N > 6 \text{ and either } p > 3 \text{ or } N \text{ is odd} \\ 2N\, D(N/2) & \text{if } N > 6 \text{ in all other cases (i.e., } p = 2, \text{ or } p = 3 \text{ and } N \text{ is even),} \end{cases}$$

where p is the largest prime factor in N and

$$D(N/p) = \sum_{j=1}^{k} p_j \alpha_j - p.$$

If N is prime and $N > 6$, this construction may require more crosspoints than an unprimed, slightly larger N.

The special case of $N = 2^n$ has been considered for permutation purposes [27,28]. An example of $N = 2^3$ is shown in Fig. 2.9. Fig. 2.9(a) shows the first iteration of Benes construction scheme and Fig. 2.9(b) shows the resulting network. In general $2n - 1$ stages are needed and the required number of 2 x 2 switching elements is equal to $\frac{N}{2}(2n - 1)$. Opferman and Taso-Wu [29] have



(a)

(b)

Fig. 2.9 Construction of Benes binary network.

(a) First iteration of Benes construction scheme.

(b) Result of Benes construction scheme - an 8 × 8 Benes binary network.

studied the control problem of this case and have shown $O(N \log N)$ steps of computation are needed if an associative memory or proper amount of memory is used.

Another interesting topic concerns the maximum number of connections which must be disrupted to establish a new connection. Some references and expansions can be found in Benes' work [26].

D. Blocking Network

A network which can perform many but not all possible connections between terminals is called a blocking network. The blocking networks are almost always found in practical designs. A survey on some specific interconnection networks of the blocking type was made by Thurber [30]. In the following we review some blocking networks which are chosen under the emphasis on the uniform structure and recent advances.

1. Four-Stage Telephone Switching Network:

The four-stage network shown in Fig. 2.10 is commonly found in telephone central offices. Benes [26, p. 123] shows that only a vanishingly small fraction of all possible permutations can actually be achieved by a four-stage network with $m = 10$ and $N = 1000$. However a rearrangeable network of $N = 1024$ which can achieve all permutations using 2×2 switching elements turns out to need 17 stages instead of 4. The simplicity and the uniform structure of the four-stage networks are attractive for the interconnection network design.

2. Series-Parallel Network:

The series-parallel networks are defined by Pippenger [31] to derive exact expressions for the blocking probability. The scheme to obtain a series connection is shown in Fig. 2.11. In Fig. 2.11, there is one link between a given primary and a given secondary. Fig. 2.12 shows the scheme to obtain a parallel connection in which there is one link between a given primary and a given secondary, and one link between a given secondary and a given tertiary. A series-parallel network is then defined as the one that can be constructed by starting with switching matrices and applying the rules

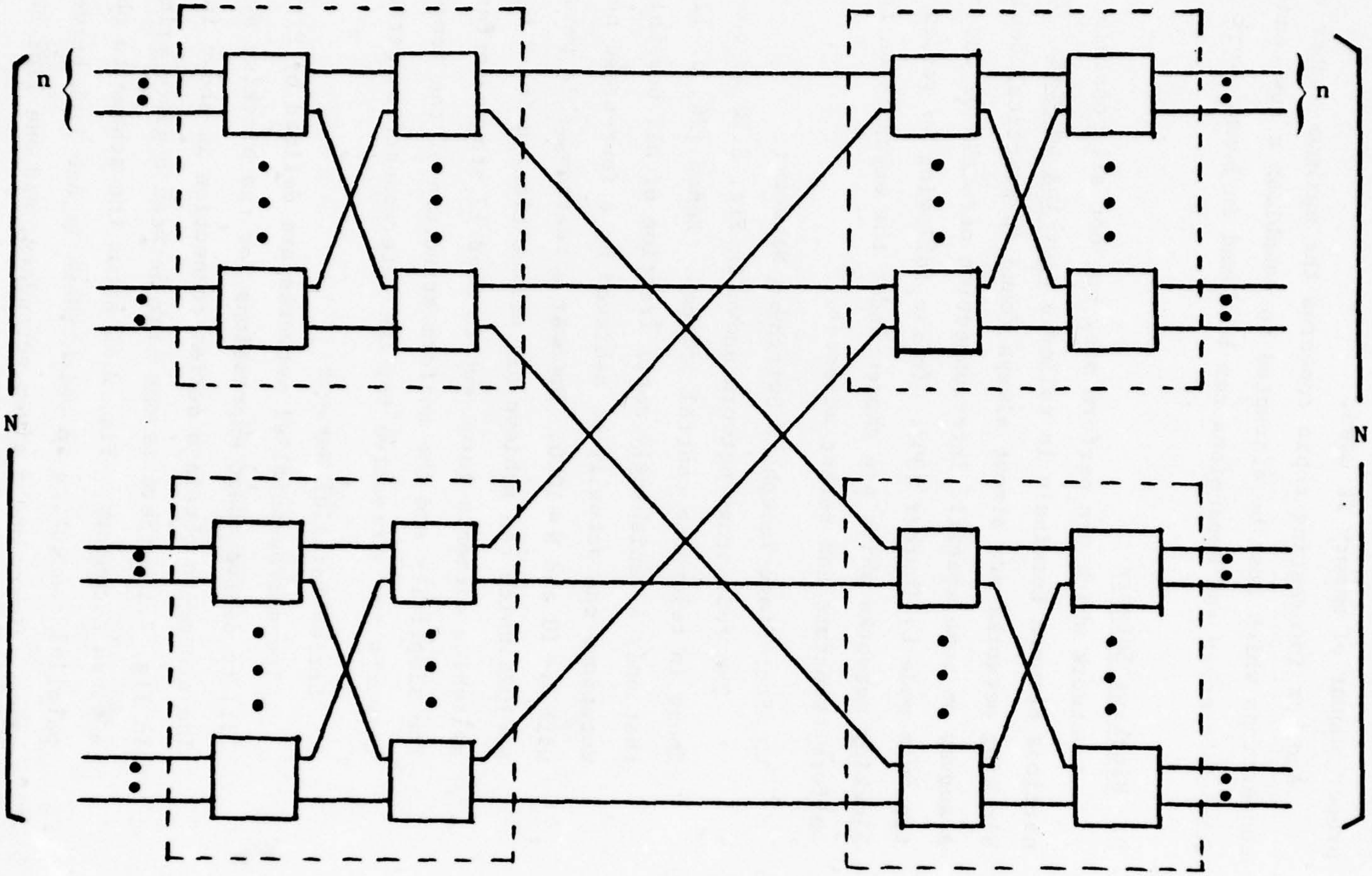


Fig. 2.10 A four-stage network which is found in many telephone central offices.

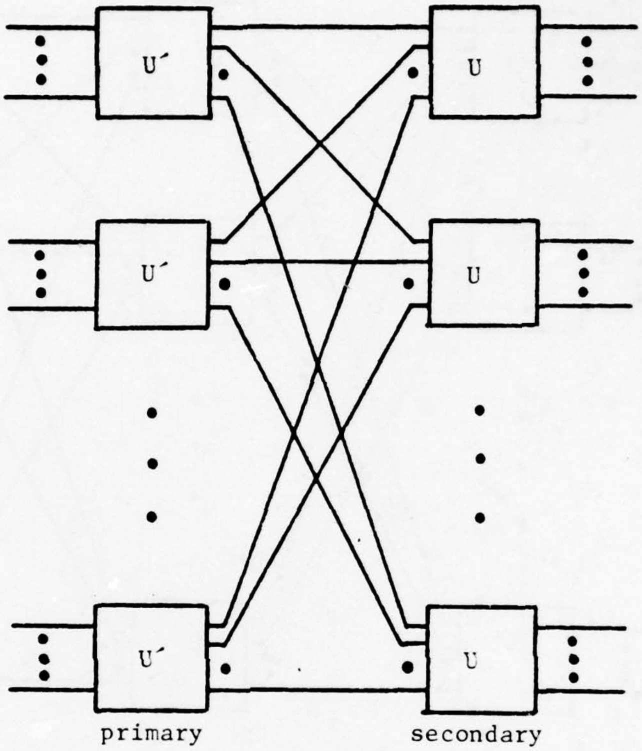


Fig. 2.11 Series connection (U´,U).

40

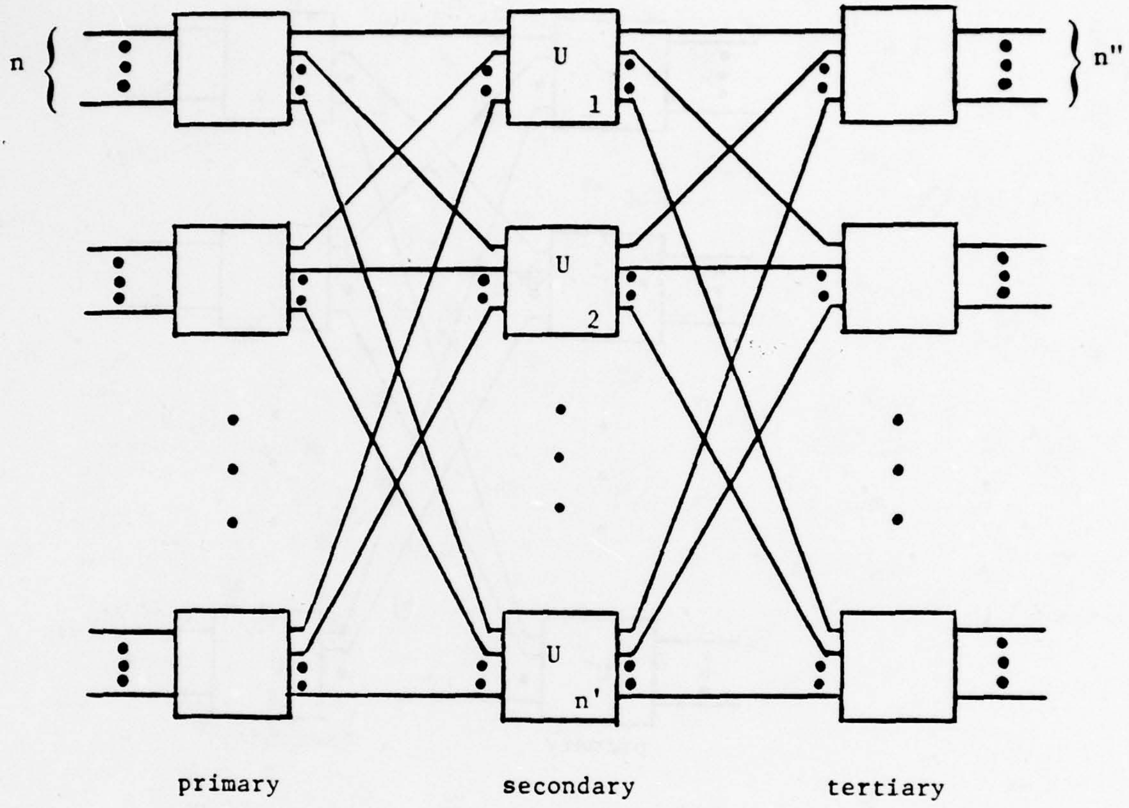


Fig. 2.12 Parallel connection (n × n',U,n' × n").

for series and parallel schemes any number of times, in any order. Pippenger concluded that networks can be constructed with less than $6N \log_2 N + O(N \log \log \frac{1}{B})$ crosspoints where B is the expected blocking probability.

3. Banyan Network:

A banyan has been defined as a tropical tree having many aerial roots that develop into additional trunks. A graph of a banyan is shown in Fig. 2.13 [30] in which there is one and only one path from any base to any apex. Lipovski and Goke specified two banyan structures [32,33] and proved a number of theorems relating to banyan networks. These two structures are SW and CC (cylindrical crosshatch) banyan structures. The SW structure recursively expands to a crossbar switch as illustrated in Fig. 2.14. The CC structure is rectangular by definition and must have S^L vertices at each level where S is the spread of the vertices and L is the level. A CC banyan network is shown in Fig. 2.15 in which $L = 3$ and $S = 2$. Some special cases of banyan networks have been proposed in a processor system architecture [34].

4. Omega Network:

Lawrie developed an algorithm to construct connection networks based on the concept of an omega base representation of integers [35,36]. Let R_n be an ordered set of integer factors of n, i.e., $R_n = (p_1, \ldots, p_k)$ and $p_1 p_2 \cdots p_k = n$, and set $\Omega(R_n) = \{W_i | W_i = W_{i+1} \cdot p_{i+1}, W_k = 1, 0 \leq i \leq k-1\}$. Using $\Omega(R_n)$, Lawrie constructs an omega network. The omega network consists of k stages (numbered 1, 2, ..., k from left to right). The ith stage is composed of n/p_i crossbar switches, each switch p_i by p_i. At each stage the inputs and outputs are labelled from 0 to $n-1$ respectively. Connections are made by connecting output j of stage i to input ℓ of stage i+1 where $\ell = (j \div W_{i-1}) \cdot W_{i-1} + (j \bmod p_i) \cdot W_i + (j \bmod W_{i-1}) \div p_i$, where $X \div Y$ is the integer part of the quotient X/Y. The input line in stage 1 should be renumbered in a special way. A network control algorithm is also provided. An omega network

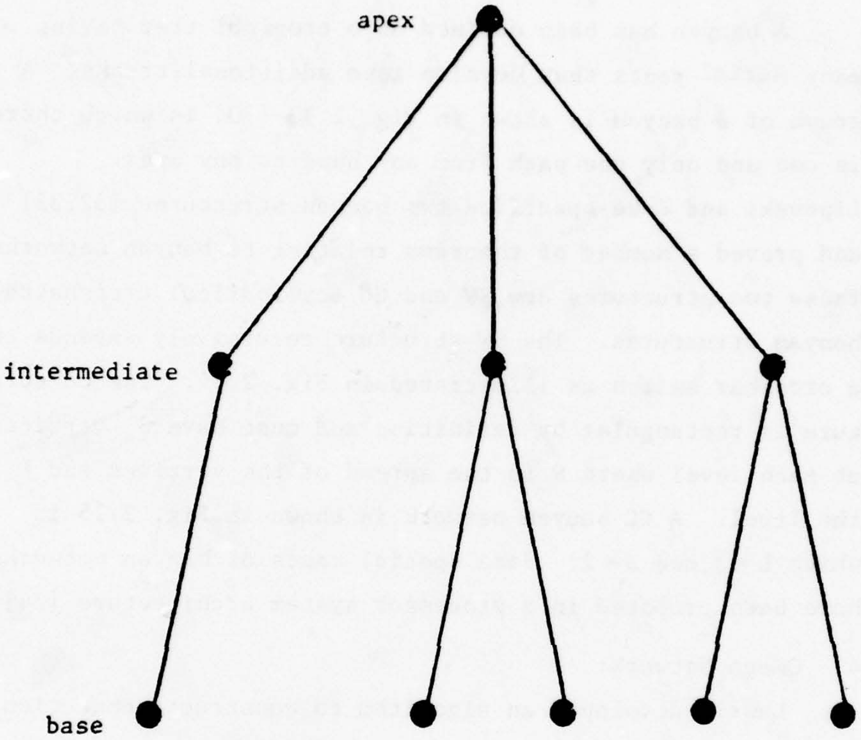


Fig. 2.13 Banyan.

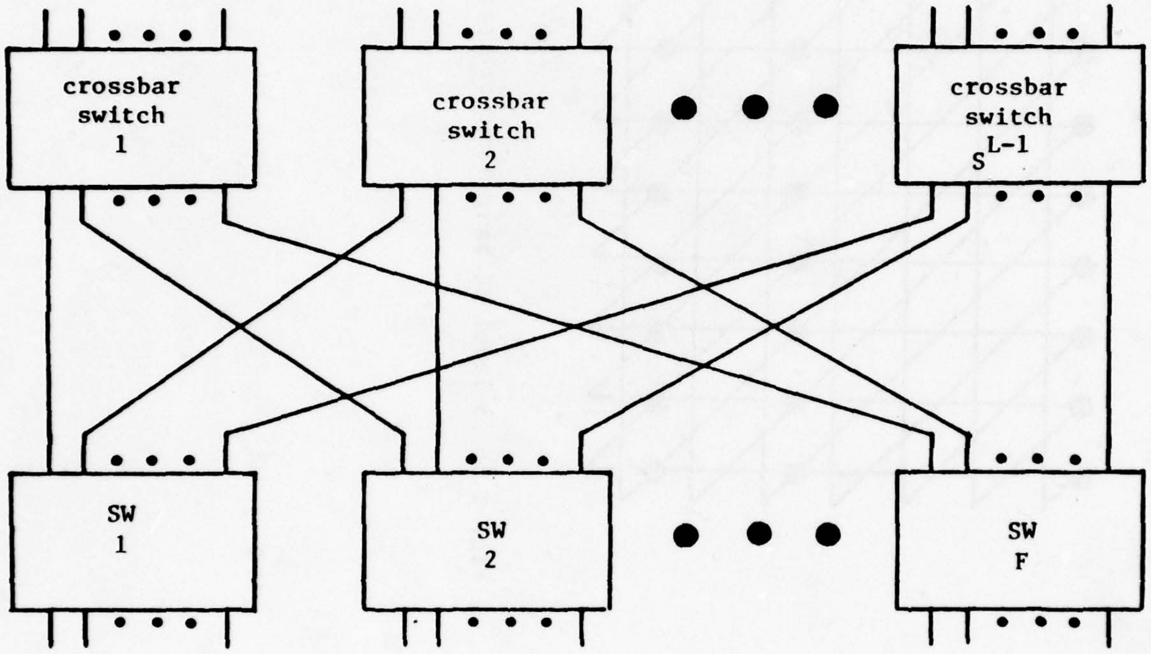


Fig. 2.14 L-level SW banyan structure with fanout F and spread S .

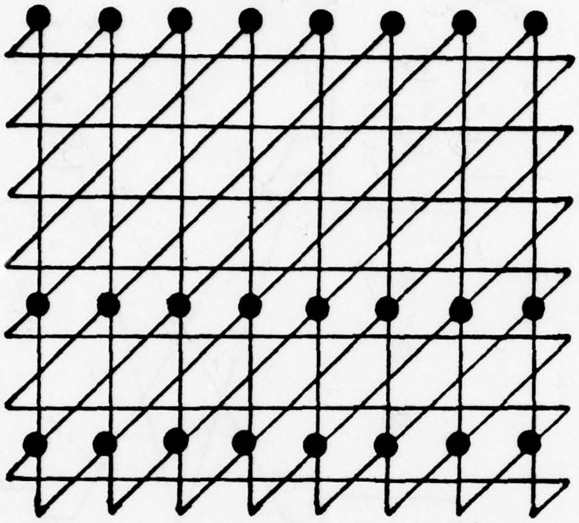


Fig. 2.15 3-level CC banyan network structure with S=F=2 .

with n = 8 is shown in Fig. 2.16.

5. Data Manipulator:

The data manipulator [37,38] is designed to achieve a set of data manipulation functions in parallel processing. There are four variable parameters in the design:

(a) the number of communication paths of each cell;

(b) the number of control line groups;

(c) the number of manipulator columns; and

(d) the interconnection paths between cells.

Appropriate control of these parameters could produce a suitable data manipulator in accordance with the specification. Figure 2.17 shows an example of three columns and six control groups. The interconnection path is such that cell X in column 2^i is connected to cell X and to the cells which differ from X in 2^i position in the adjacent column. Thus the required number of stages is $\log_2 N$. The number of communication paths shown in Fig. 2.17 is equal to three. The six control groups are shown in Fig. 2.18.

6. Versatile Line Manipulator:

The versatile line manipulator is capable of achieving almost all the data manipulation functions [39]. Figure 2.19 shows a block diagram of the versatile line manipulator.

There are N × N cells (Fig. 2.20) in the basic line-manipulator circuit (BLMC). The output gate (OG) of cell (i,j) is controlled either by the ith address control register (through a decoder) or by a combination of the input and output control registers. The ith address control register determines the location of OG in the ith BLMC row to be activated. Thus, only one OG in each BLMC row can be activated at a time while there could be up to N OG's activated in each column by providing the same addresses for the address control registers. A versatile data manipulator has been implemented in STARAN [40].

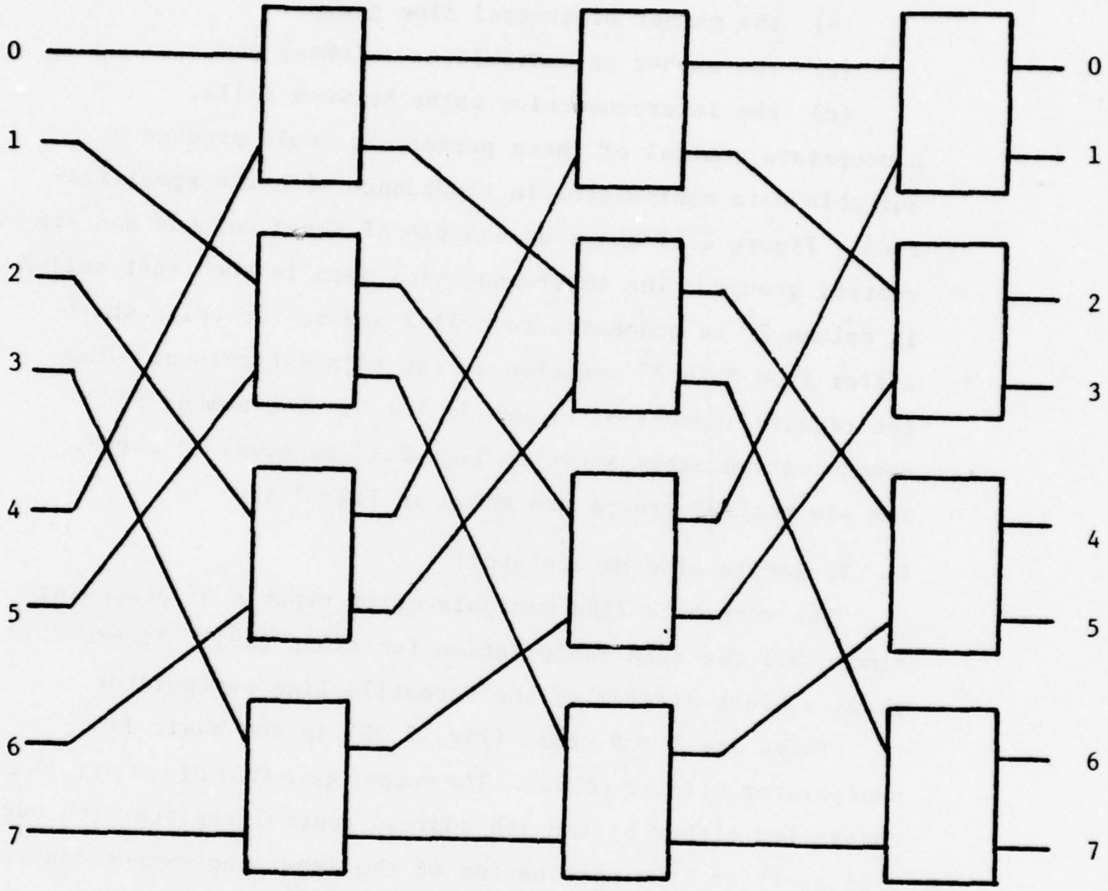
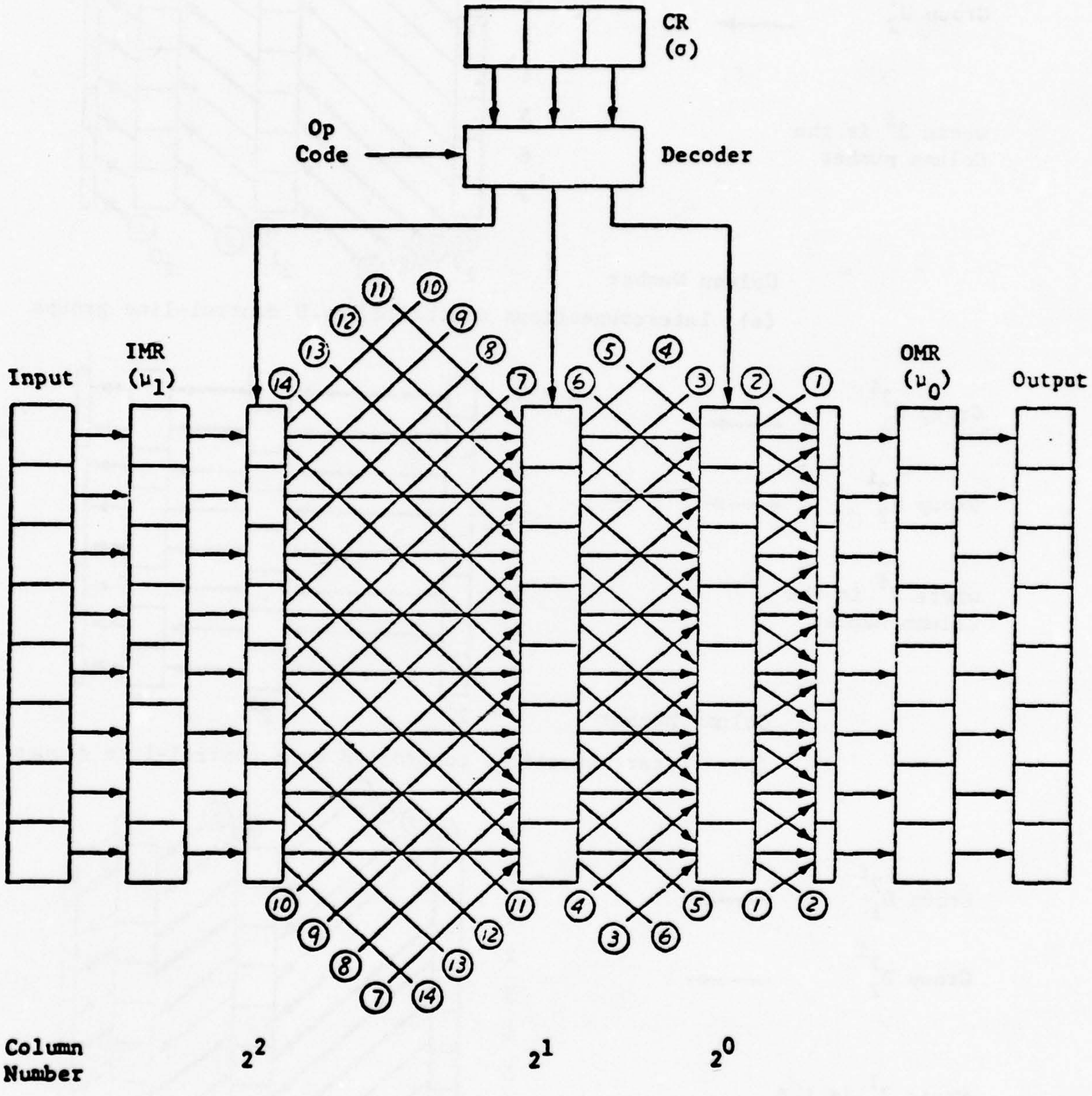
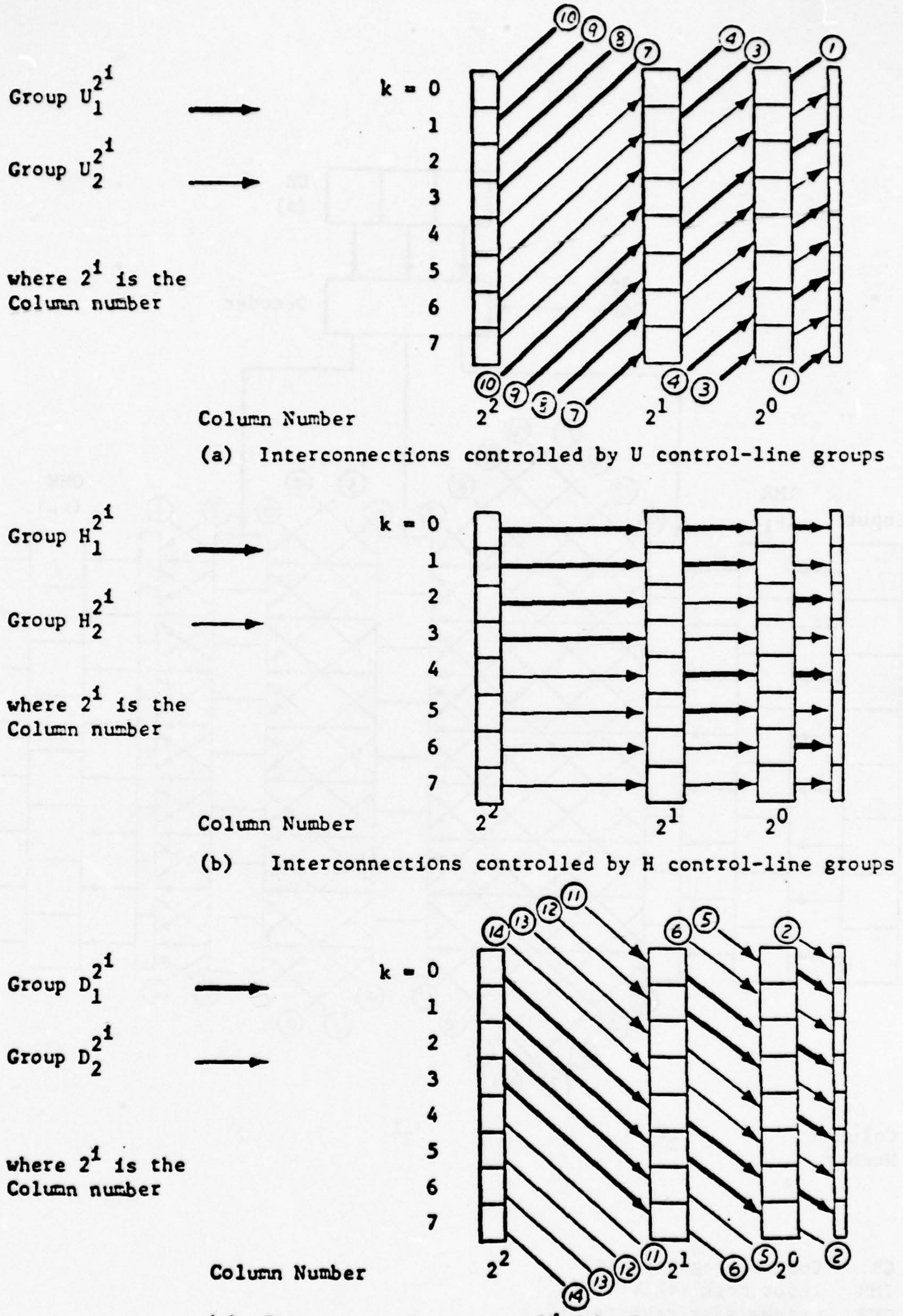


Fig. 2.16 An omega network for n=8



CR Control register (σ)
IMR Input mask register (μ_1)
OMR Output mask register ($\bar{\mu}_0$)

Fig. 2.17 A data manipulator for 8 items.

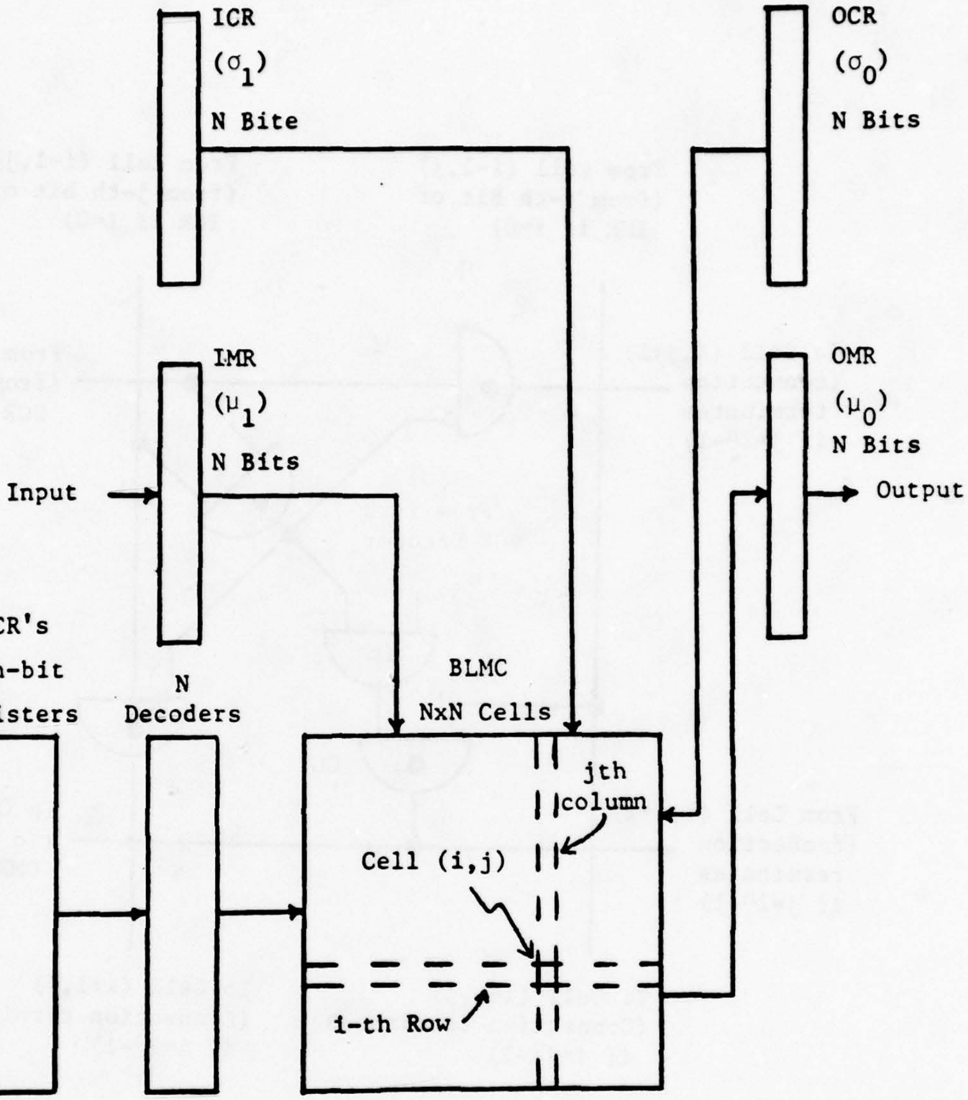


(a) Interconnections controlled by U control-line groups

(b) Interconnections controlled by H control-line groups

(c) Interconnections controlled by D control-line groups

Fig. 2.18 Control group for the data manipulator.



ACR	Address Control Register
BLMC	Basic Line-manipulator Circuit
ICR	Input Control Register (σ_1)
IMR	Input Mask Register (μ_1)
OCR	Output Control Register (σ_0)
OMR	Output Mask Register (μ_0)

Fig. 2.19 A versatile line manipulator (VLM).

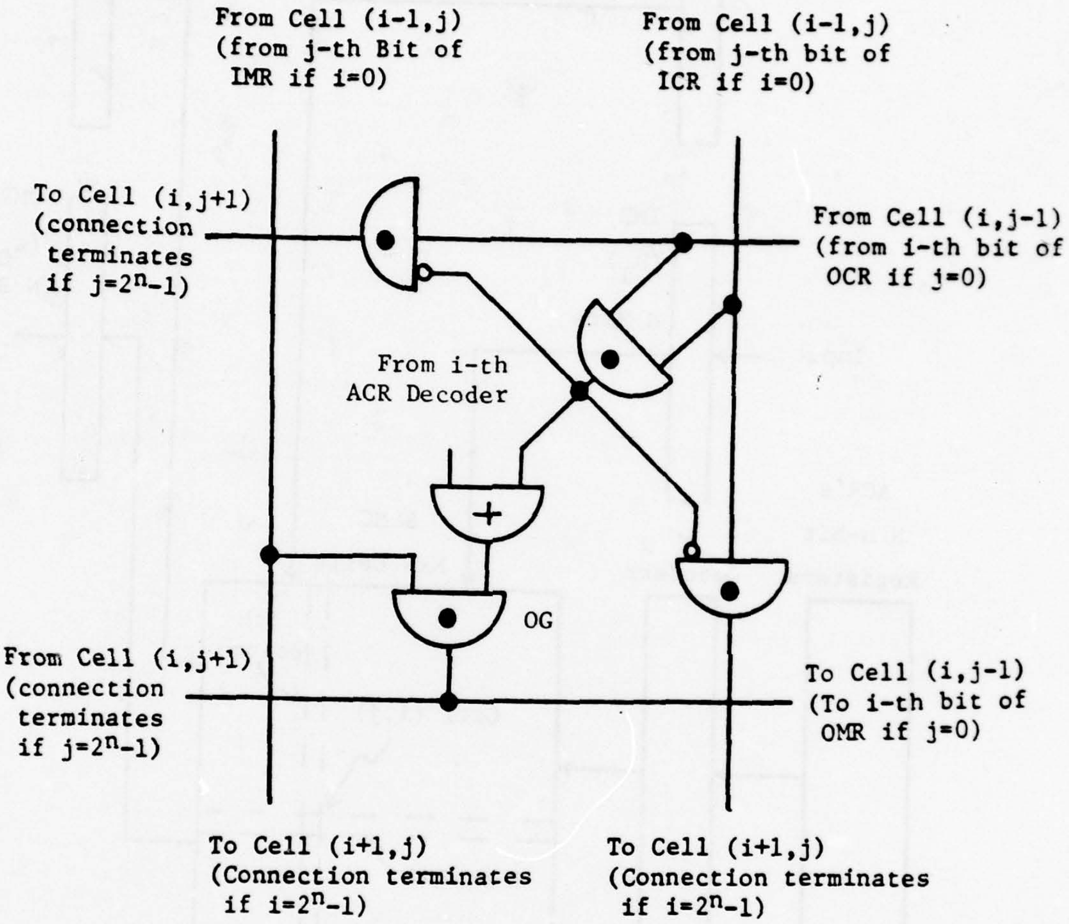


Fig. 2.20 **The logic circuit of BLMC Cell (i,j).**

7. Expanded Shuffle-Exchange Network:

The perfect shuffle permutation was used by Pease [41] in 1968. Stone [42] constructed a folded one-stage network which can perform an exchange and a shuffle function. In a later paper a simple control mechanism is added to allow some permutations [43]. The control scheme requires only one bit per cell and the bit value is determined by an EXCLUSIVE OR operation of the bit values in the previous step. The shuffle-exchange functions have been found useful in some processing applications. An expanded multistage shuffle-exchange network is shown in Fig. 2.21.

8. Flip Network:

The flip network [19] scrambles and unscrambles data for the MDA memory [44]. It can also perform the processor-to-processor routing required for many problems. Fig. 2.22 shows an 8-item flip network for flip permutations. Each stage is controlled by a control bit. If the control bit is 1 the stage performs crossed connecting, otherwise the direct connection is performed. A network shown in Fig. 2.23 with modified control strucrture can perform some shift permutations.

9. Indirect Binary n-Cube Network:

The indirect binary n-cube network is used to permute data in a microprocessor array [45]. The basic form of the indirect binary n-cube is illustrated in Fig. 2.24 for $n = 4$, $N = 2^4 = 16$. The circles in Fig. 2.24 represent the microprocessors, indexed from 0 to $(2^n - 1)$ as indicated by the numbers in the circles. The lines on the right from the switching network connect back to the microprocessors with the indices given in parenthesis. The network uses switching elements with direct and crossed connection capabilities and allows multiple passes to obtain permutations of data. A control system using global command which can be broadcast to all microprocessor and a set of switch controllers is suggested.

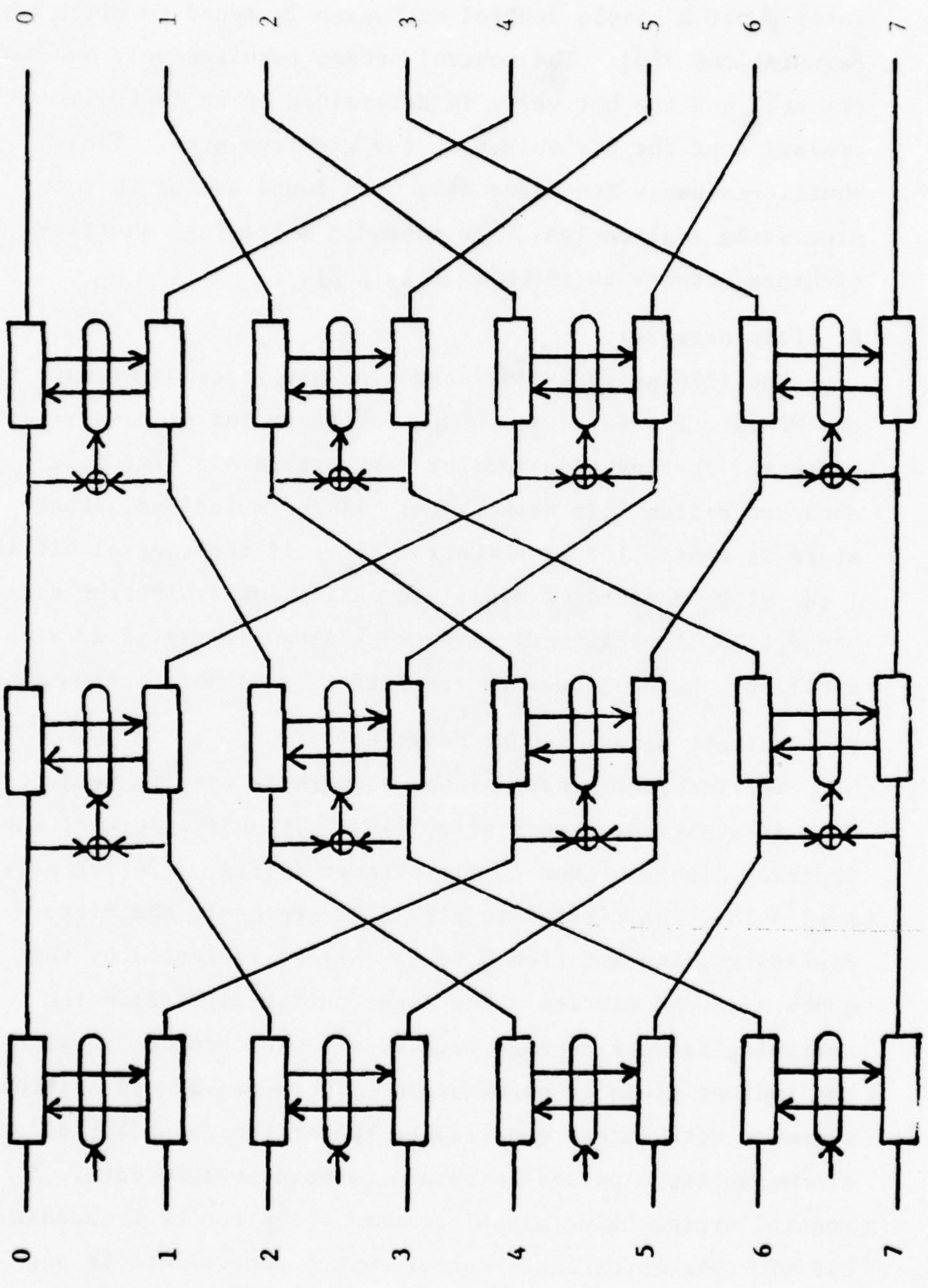


Fig. 2.21 An extended shuffle-exchange network.

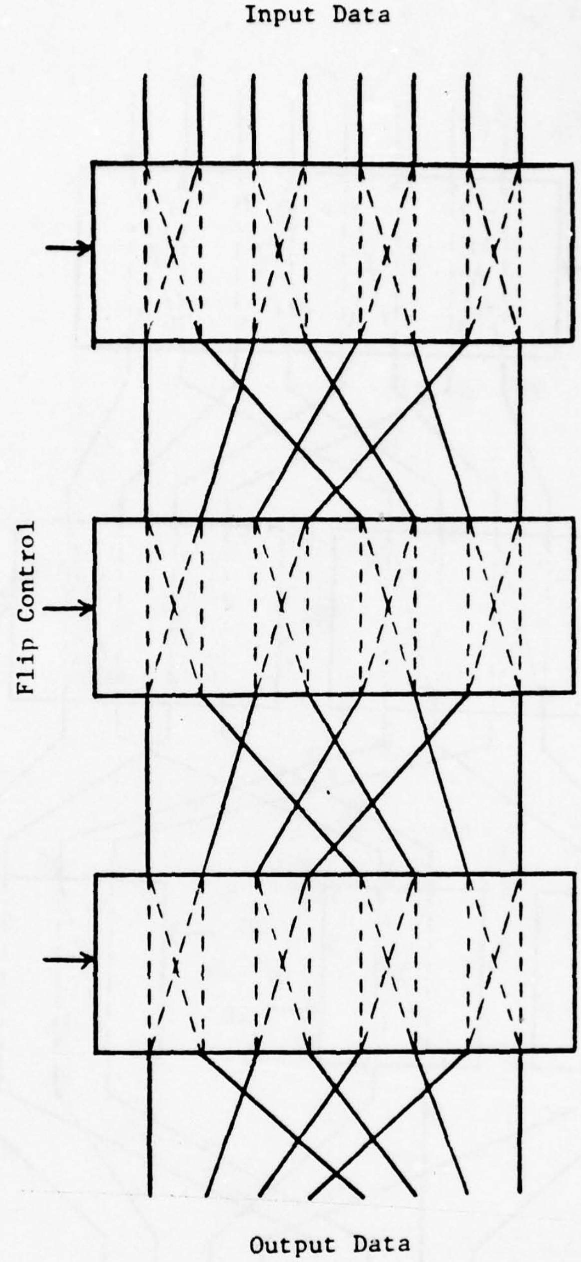


Fig. 2.22 An 8-item flip network .

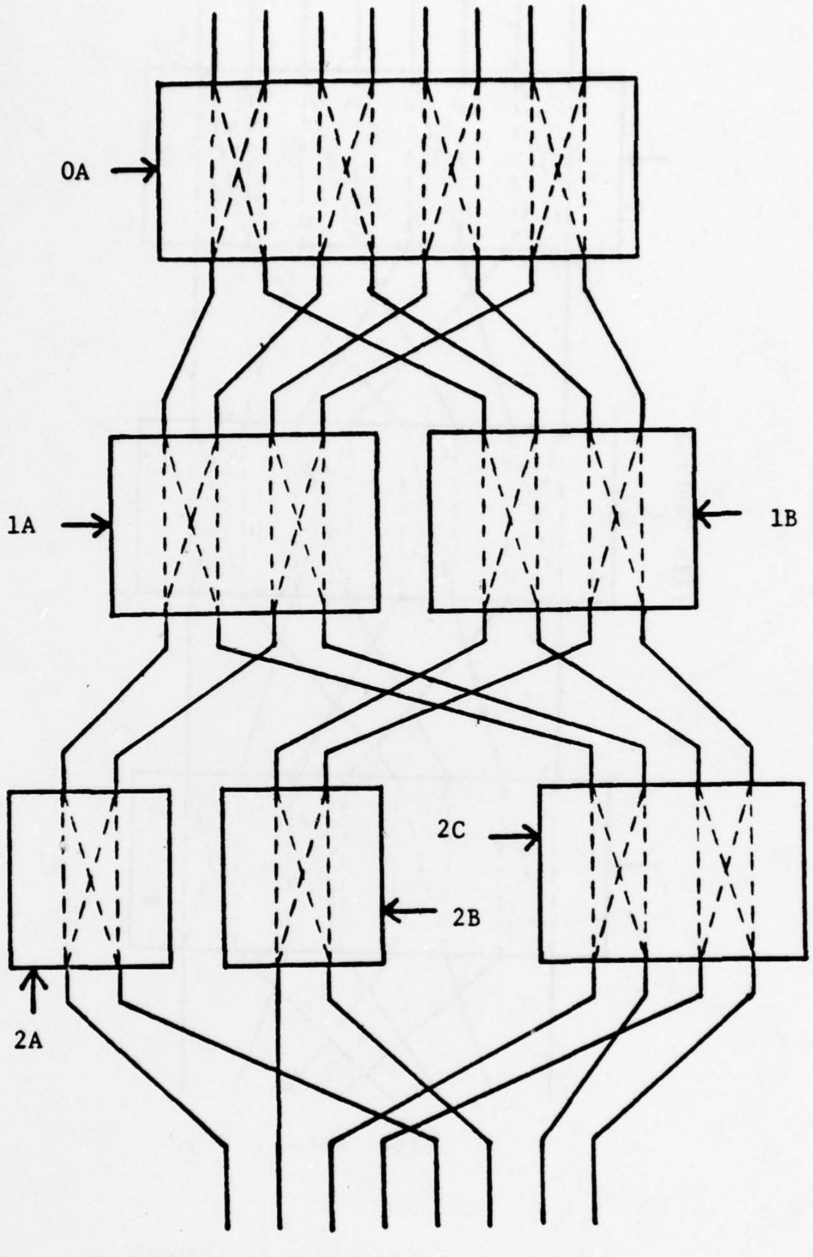


Fig. 2.23 An 8-item network for flip and shift permutations.

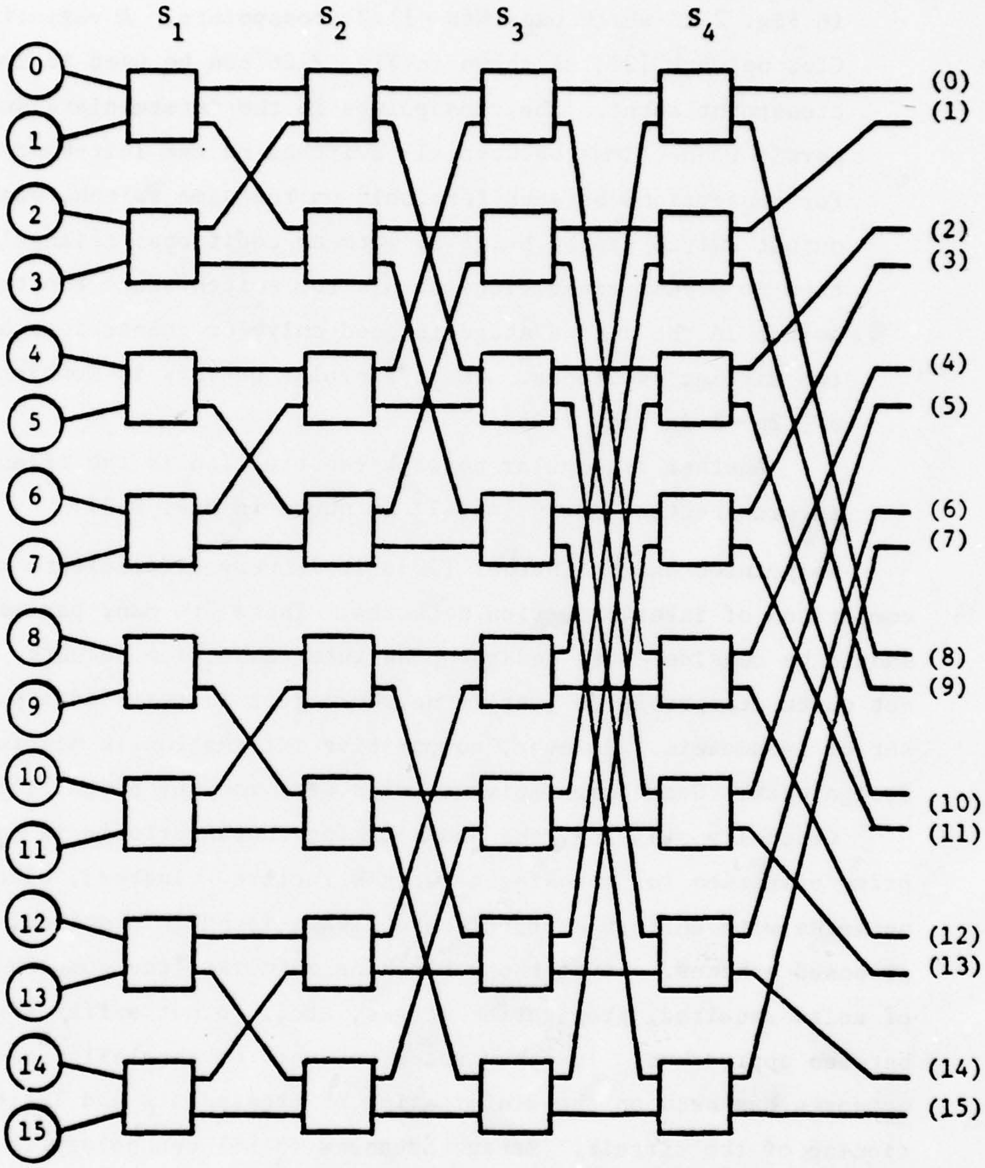


Fig. 2.24 The indirect binary 4-cube network.

E. Triangular Network

For the one-sided case where both inputs and outputs are on the same side, a triangular array as shown in Fig. 2.25 can be used for construction. The straight way to construct a nonblocking triangular network of size N is to use the triangular array shown in Fig. 2.25 which uses N(N - 1)/2 crosspoints. A variation of the Clos network [22] as shown in Fig. 2.26 can be used to improve the crosspoint count. The crosspoints in the intermediate switches permit connections between all switches on the left-hand side. For connections between terminals on the same switch, each input-output switch can be built up with an additional triangular portion to permit connections within the switch since the triangular switch in the second stage is good only for connections between two distinct switches. The triangular network is nonblocking for $m \geq 2n - 1$ in Fig. 2.26

Another triangular network construction is the triangular interconnection array [46,47] as shown in Fig. 2.27.

As pointed out by Thurber [30], it is very difficult to make a comparison of interconnection networks. There are many parameters that should be considered in designing an interconnection network. It is not quite acceptable to justify networks just in terms of a partial set of parameters. However, comparative information is helpful to design work. Some comparative studies were mady by Siegal [48-50].

Generally speaking, the non-blocking characteristic is not the prime criterion for choosing network structure. Instead, blocking networks with uniform structure are always found in practical usage and proposed schemes. Among those blocking networks, the complexity, number of units required, propagation stages, etc., do not differ significantly between approaches. In other respects, work on complexity analysis of networks has been on the minimization of crosspoints and logic partitioning of the circuit. Recent advances in LSI technology relatively cut down the merit of this minimization criterion and also lead to new computer architecture trends. With this background, one should consider some new design concepts which are described in the following section.

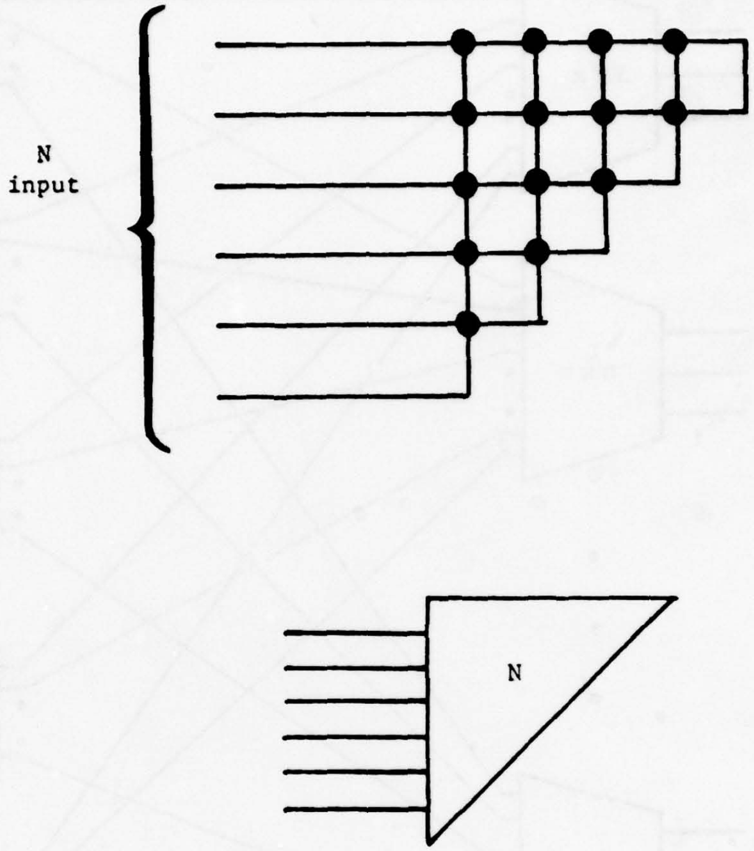


Fig. 2.25 A triangular array and its symbol.

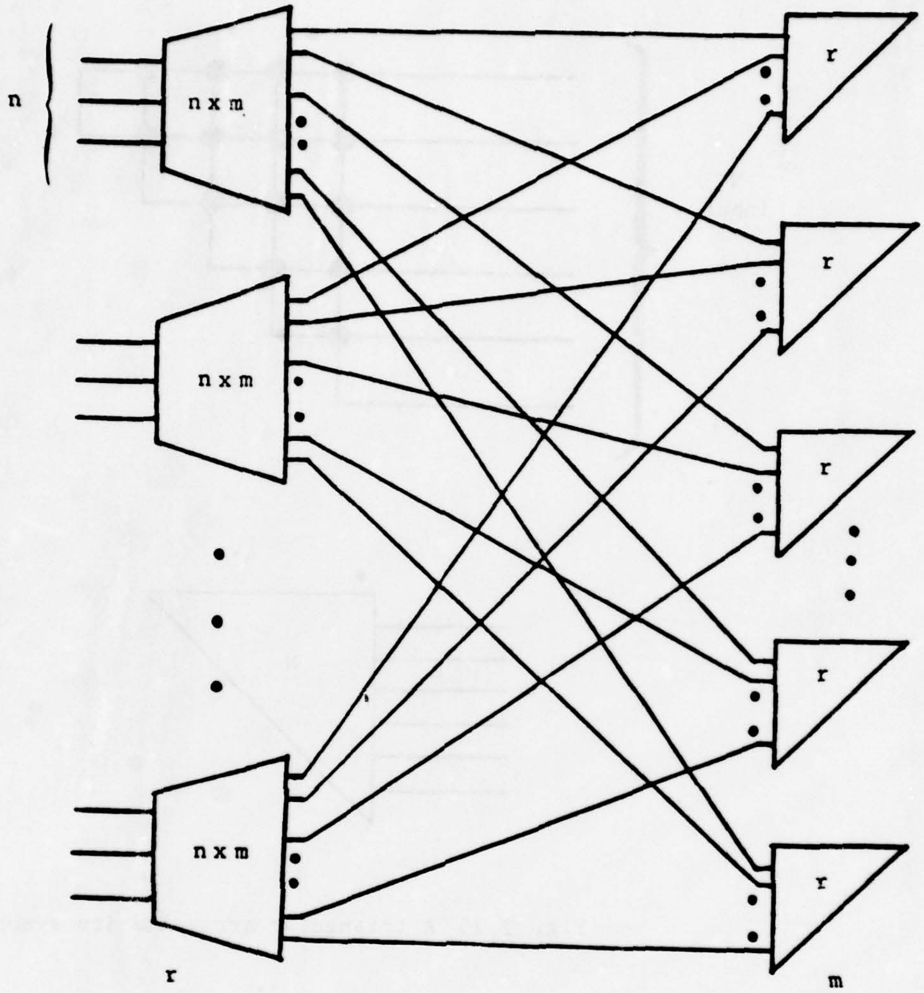


Fig. 2.26 Clos construction for a one-sided triangular network.

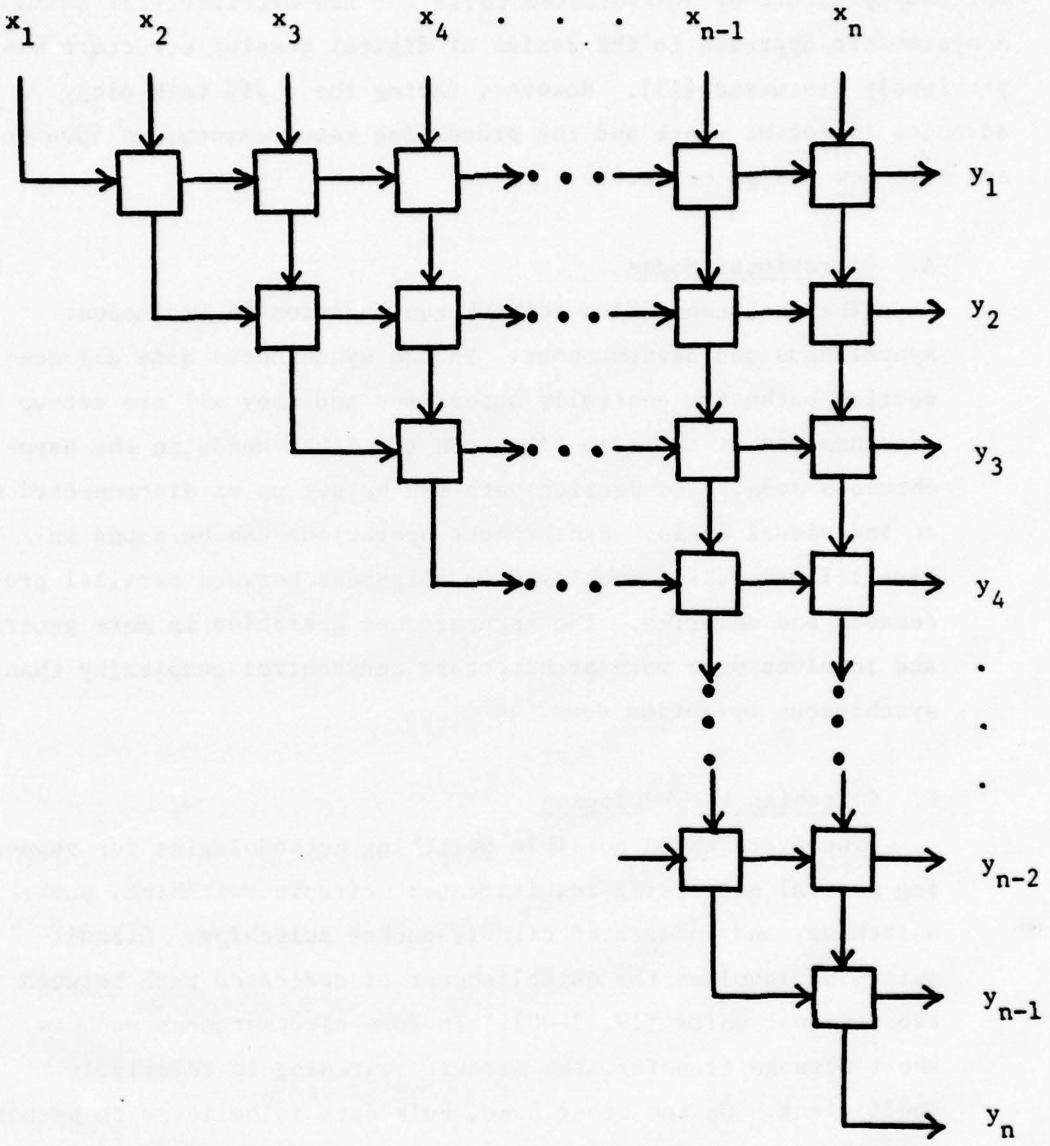


Fig. 2.27 A triangular interconnection array.

60

2.3 General Design Criteria

As the complexity of computer systems grows because of the technology advances and the processing requirements, the interconnection philosophy should be re-evaluated tofit the new architectural trend. A systematic approach to the design of digital bussing structure was previously discussed [15]. However, facing the rapid technology advances in recent years and the processing requirements, we have to add some new design criteria.

A. Operational Modes

The interconnection network can function in two modes: synchronous and asynchronous. In the synchronous mode all connection paths are centrally supervised and they all are set-up and disconnected at the same time. On the other hand, in the asynchronous mode, a connection path can be set up or disconnected on ar individual basis. Synchronous operations can be found in parallel processing such as data alignment between parallel processors and memories. The asynchronous operation is more general and involves much more architecture and control complexity than the synchronous operation does.

B. Switching Methodologies

There are three possible switching methodologies for supporting general networking requirements: circuit switching, packet switching, and integrated circuit-packet switching. Circuit switching involves the establishment of dedicated path between two terminal units [19,32-40]. In some circumstances such as short message transfer, the circuit switching is relatively inefficient. On the other hand, bulk data is believed to be more suitable for circuit switching. In contrast to circuit switching, packet switching attempts to multiplex the use of the communication circuit among all related terminal units [51,52]. Messages are typically broken into a series of fixed length, addressed packets of data which are routed independently to their destination using store-and-forward procedures. The packet switching can partially solve the blocking problem of the interconnection network. However, it also increases the complexity of the routing procedure and the cost of the system. A compromise solution for the short message and the bulk data is the integrated circuit-packet switching.

C. Routing Techniques

The routing techniques of the intercommunication subsystem in a large multiple-processor system is no longer simple as those for bus lines. The technique involves computing the available path, and resolving conflicts, etc. Essentially three techniques have been considered: centralized, distributed and adaptive. In a centralized routing scheme [29], a central authority takes over all logic decisions which are needed to set up the intercommunication. One of the disadvantages [52] of centralized routing is that the central authority becomes very difficult to implement when the number of processors increases tó as large as 2^{14}. A local distributed routing technique may be used to overcome this disadvantage. However, there is no coordination among the local decisions and hence the resource in the subsystem may not be used efficiently. An adaptive scheme can then be implemented by collecting the subsystem status information and making routing decisions locally.

D. Communication Techniques

The communication techniques become more critically important since there could be more than one decision point along a path set up in an interconnection network. There are several protocol communication levels such as: link control, switching element-to-switching element, and terminal-to-terminal. The synchronization problem is a difficult one. Various techniques [53,54] should be carefully reviewed in order to tackle the problem more effectively.

E. Interconnection Network Structure

The system performance and cost of intercommunication subsystems largely depend on the interconnection network structure. The time-shared or common buses cannot fit for the system with a large number of processors and memory modules. Crossbar switches have two disadvantages as described in Section 2.1. The multiport memory scheme can also cause large circuit components if the number of the processor units becomes large. The multistage interconnection network represents a feasible approach from both the functional and cost points of view. However the multistage network should be partitioned in such a way that the building block modules are cost effective for LSI implementation and software development.

F. Interaction Level

One of the important questions is what interaction level the interconnection network should have in relation to other subsystems. A preferred design in the distributed processing [8] is that the interconnection network becomes an autonomous subsystem and interacts with other subsystems by message transaction. The unit of message transaction may be set as a file, a data set or an instruction.

G. Design Tools

For a large interconnection network the performance can be improved through the analysis of the entire system [55]. Some parameters such as the message size, number of buffers, etc., may be set arbitrarily and get tuned to optimized values thereafter. The tools for the analysis should be planned and incorporated into the system.

H. Others

Some other decisions should also be made for designing intercommunication subsystems. These include traffic classification, serial or parallel transmission, digital or analog signals, fault diagnosis scheme, correcting structure, etc.

2.4 Characteristics of Interconnection Networks

To facilitate the communication between the user and the designer and to set a goal for the design, a set of specifications on the interconnection networks should be developed. Some characteristics which can be used to specify the interconnection network are identified as follows:

A. Topology of Interconnection Network

The topology includes the communication paths of each switching element, the connectivities of the communication links and the number of switching-element stages.

B. Control Structure of Interconnection Networks

The control structure can be classified into three categories [50]: individual stage control, individual box control, and partial stage control.

C. Logical Complexity

This refers to the totality of decisions made during communication. It should be as small as possible.

D. Blocking Probability

This quantity is used to measure the probability with which an intercommunication subsystem responds to transfer requests from the terminal unit.

E. Message Response Time

There are two meaningful measures on response time: terminal response time and overall response time. The terminal response time is defined to be the time required from the instant the transmit is sent to the moment the reply message begins to appear at the terminal. The overall response time is the elasped time from the instant a message arrives at a terminal to the moment the message is completely served.

F. System Capacity or Throughput

The capacity is defined as the maximum traffic that a system can carry, while satisfying the blocking probability criterion. and/or response time requirements.

G. Network Reliability

The reliability can be defined as the mean time between two failures (MTBF) and the average man hours required for a repair.

H. Sensitivity

This is the effect that the system would experience if the actual traffic is above the project or if some tolerant faults appear in the network.

I. Traffic Bottleneck or Deadlock

This is an inherent performance limitation due to a non-uniform flow of communication or to saturation of a shared resource.

J. Transmission Error Rate

The error rate is a function of message size, line condition and hardware characteristics. It should be calculated in order to decide whether the error correction circuit is needed.

K. Cost

There are tradeoffs for cost/performance. Technology advances allow design alternatives. A set of curves to weigh and compare the tradeoffs can be developed. It is the designer's responsibility to design a least cost network while satisfying the requirement.

2.5 Network Hardware and Software Design Issues

There are mutual relationships among the network hardware and software design, the choice of network design decisions and network specifications. Since the factors affecting the hardware and software design vary case by case, we discuss only the general issues in this section.

A. Network Hardware Design

Some hardware features associated with switching matrix, network control, linkage and connectivity, synchronization, and functional unit-network interface are discussed.

1. Switching Matrix:

The required features in a switching matrix can be classified into two parts: controller and transfer unit.

a. Controller: The complexity of a controller largely depends on the control technique and the switching method. The simplest is probably the one which uses the circuit switching and the central control technique. In this case the switching matrix receives control signals from the control center and stores them in registers to set up the connection path in the transfer unit. Another possibility is to use the circuit switching and the distributed control technique. The controller should be able to receive destination information from its neighbors, decode it, resolve conflicts and dispatch proper control signals to the transfer unit. Obviously the controller should have the logical decision ability and some memories should be included for conflict table usage. The controller for the packet switching is not less complex than the one previously described. Some important issues include the buffer management and the routing table updating. Another important aspect is the reliability consideration. A self-fault-detection mechanism should be installed to make sure that the switching matrix is properly functioning, or otherwise inform the fault to the neighbors and the global network control center. Since the dynamic decision capability should be built into the controller to fulfill the requirements of the distributed control, the inexpensive microprocessors provide a great potential for this need [56-58]. In the near future the dynamic distributed microprocessor switching matrix will certainly replace the static switching matrix in the interconnection network. However, the logical complexity should be made low and the size of the switching matrix should be properly chosen in order not to overload the microprocessor controller.

b. Transfer unit: In the circuit switching mode the transfer unit is a connection network in which a connection path can be established between an input/output pair. In the packet switching mode, the transfer unit represents the buffer pool system. Some connection networks have been implemented [19,37-40] and some have been proposed [34,42]. In any circumstance, the connection network should be designed in such a way that the controller and the fault diagnosis mechanism can be relatively simple. The buffer pool system can be implemented by high speed memory. However, the DMA capability of that memory and the data structure of buffers are critical factors involved in time delay.

2. Network Control Unit:

There are several reasons that a network control unit should exist. Among the reasons are the central control for the circuit switching operation, reliability considerations and network reconfiguration, initialization of switching matrix, and performance measurement. Opferman and Tsao-Wu have described hardware requirements of a network control unit for a rearrangeable network [29]. For a large interconnection network operating in the circuit switching mode, the central control unit can be as complex as that for No. 1 ESS [59] which consists of dual processors, temporary and semipermanent memories, scanner and distribution units. The network control unit should also be able to collect (or test) the fault information and provide network reconfiguration by updating the local routing tables. The switching matrix initialization and the performance measurement should also be controlled by the network control unit.

3. Linkage and Connectivity:

The links between switching matrices provide the connectivities between terminals. For the graceful degradation reason, there should be at least two disjoint connection paths existing between a terminal pair. The links may be used to transfer not only data but also control signals and can be designed as full duplex, half duplex or simplex lines.

4. Synchronization:

There are no significant timing or synchronization problems in an analog communication network. However, synchronization plays an important role in a digital communication network. The synchronization affects not only the netwoık reliability but also the message throughput. It is possible to use master-slave synchronization schemes in the interconnection networks.

5. Interface between Functional Units and Network:

Basically the functional unit-network interface is not a part of the network under design. However, its input/output format affects the functioning of the network. A sophisticated interface should be used to handle the interactions between a functional unit and the network.

B. Network Software Design

It is essential to have a set of basic routing procedures to insure an efficient, correct and smooth transfer of information in the interconnection network system. In general this basic routing procedure can be partitioned into four categories [60, 61].

1. Communication Protocols:

A communication protocol is a set of rules established to manage the information exchange between two terminal units. The protocol allows the terminal units to understand one another and to cooperate with one another. A good communication protocol can result in short delay time and high throughput. Usually protocols can be classified into several levels.

2. Flow Control Procedures:

Flow control procedure [62] regulates the input amount and rate that an interconnection network can accept in order to prevent or minimize the occurrence of traffic congestion and deadlock. There are two problems which should be worked out in the flow control procedures. The first one is to minimize the occurrence of congestion and deadlock. The second one is to handle the congestion and the deadlock if they unfortunately occur. In circuit switching mode, the connection request can be abandoned or deferred if this connection request should result in congestions. In packet switching mode, the buffer regulation scheme of the computer communication can be used. There are two existing congestion control methods [63] for packet switching,which can be classified as "local" and "end-to-end". An overflow buffer scheme is used to thwart the deadlock in the packet switching of computer communications. Careful study should be made to apply the developed schemes in the newly investigated multistage interconnection network system.

3. Graceful Degradation Considerations:

A factor of major importance to the successful operation of the interconnection networks is their reliability [64,65]. The effect of failure in the switching modes and links should be minimized. Indeed a high reliability can be achieved by having at least two disjoint paths between every pair of terminal units. To facilitate this multiple disjoing path, multiport processor units should be developed. However, like the duplication reliability scheme in the telephone switching system, the multiple disjoint paths scheme proceeds under the assumption that there is a fault detection and recovery process existing in the network system. A network control unit is preferred in addition to the self-detecting mechanism. The detection of fault should induce a recovery process such as rerouting, data restoration and failure replacement messages.

4. Routing Procedure of an Intercommunication Network:

This is one of the factors which can influence communication delays and traffic throughputs. In circuit switching, the routing procedure establishes a dedicated circuit line between source and destination and the data flows through the established line. In packet switching, data is routed from source to destination without establishing a dedicated line. In both switching modes the purpose of the routing procedure is to send data from source to destination. The philosophy for the routing procedure in both switching modes is the same although the implementations are different. Some potential candidates for routing procedures are discussed as follows:

a. Centralized routing procedure: This procedure utilizes a central authority which dictates the routing decisions [29,36,38,45]. The central authority processor collects the network status information, makes optimal route decisions and dispatches control signals to the network.

b. Distributed routing procedure: In this procedure the required processing for the routing control is distributed over processing elements within the network with several elements executing tasks concurrently. This method could solve the capacity limitation in the centralized routing procedure. Some examples are listed as follows:

i. Saturation routing - The saturation routing means a flooding of the network with messages which search for a particular identification number over every possible path. In the saturation routing, a search causes all unconnected paths to be unavailable to other simultaneous searches. This disadvantage could cause high blocking probability and hence large delay time.

ii. Hamming distance decreasing routing - An addressing scheme has been proposed for ring communication network using ternary numbers [66]. It uses a distance matrix to encode every switching element in such a way that the Hamming distance between two code points is equal to the distance between the correspondent switching elements. It has been shown that the addressing scheme is good for any network structure. The route can be formulated as follows: If the search message arrives at switching element A, calculate the Hamming distance between the destination and A, and the Hamming distance between the destination and switching elements adjacent to A. The message is routed to the switching element which is Hamming distance 1 closer to the destination than is A. One of the disadvantages is that the binary code for the ternary address is too long to handle if the network is of a large size.

iii. Table look-up routing - A compromise solution to the disadvantage in (ii) is the table look-up method. In this method, each switching element maintains a destination table. The message can be routed according to the table contents. Some adaptive routing method can be implemented by updating the table according to the network status information [67,68]. Two of them are isolated adaptive routing and distributed adaptive routing.

iv. Destination tag routing - If the location of the switching element is not restricted by the geographical factors, we can design a network structure from which we can take advantage of its regularity for developing a routing procedure [35]. This routing scheme should be extended to more general cases.

It might be necessary to combine several routing schemes in a routing procedure.

2.6 Summary

In this chapter, we have reviewed the interconnection organizations of multiple-processor systems and classified them into four categories. Among these four interconnection organizations the multistage interconnection networks have been receiving increasing attention because of the impact of recent LSI advances. After reviewing some specific multistage interconnection networks, we provided the general design criteria, some characteristics of interconnection networks, and the hardware and software requirements for designing an intercommunication subsystem for multiple-processor systems. It is noted that designing an efficient intercommunication subsystem for multiple-processor systems is one of the most important challenges in today's computer architecture.

CHAPTER 3

A CLASS OF MULTISTAGE INTERCONNECTION NETWORKS

Some multistage interconnection networks have specifically been proposed or implemented during the past several years. These networks include the data manipulator (modified version) [38], the indirect binary n-cube network [45], the flip network [19], the omega network [35,36], and the regular SW banyan network with spread and fanout of 2(S=F=2) [32-34]. Each of these networks consists of a set of N input terminals, a set of N output terminals, $\log_2 N$ stages of logic cells, and a set of control lines. The set of N input terminals and the set of N output terminals are two disjoint sets of terminals. All of these networks are capable of connecting an arbitrary input terminal to an arbitrary output terminal. But, simultaneous connections of more than one terminal pair may result in conflicts in the communication path within the logic cells.

This chapter investigates the practical nature of the interconnection networks by comparing the existing interconnection networks and applying them to the real-world problem. In Section 3.1, we define a baseline network which can serve as a reference for evaluating the relationships among the most existing interconnection networks and compare the baseline network to the existing networks mentioned in the previous paragraph. Section 3.2 describes a simple routing procedure which can result in the same connection path no matter which side of the terminals is chosen as the input side and the other side as the output side. The routing procedure also includes a scheme for the conflict resolution. Section 3.3 extends the routing procedure to allow one-to-one connections between all pairs of terminals.

3.1 Isomorphic Topology

In the first part of this section a baseline network is introduced. Then a topological equivalence of several multistage interconnection networks is given.

A. Configuration of a Baseline Network

The performance of an interconnection network is determined largely by its configuration. By the configuration of an interconnection network we mean the topology and the label of the components of the network. Here we define the topology in terms of three of the four variable parameters for designing a data manipulator [38]. These three parameters are the number of communication paths of each switching element, the number of columns (or stages) and the interconnection paths (or links) between switching elements. In this chapter we consider $\log_2 N$ stages of 2×2 switching elements, i.e., switching elements each with two input and two output terminals, and describe the connectivities of the interconnection paths between switching elements using a set of mathematical rules, called topology describing rules, derived directly from the structure definition of the network. Fig. 3.1 shows a 2×2 switching element which has capabilities of direct and crossed connections. The logical names of the components of a network can be used to unambiguously identify each link and each switching element in the network by applying some mapping rule on their physical names. A physical name is given to each switching component (stage, element, link) for identifying its relative location in the network in order to describe its topology.

Now we introduce a baseline network which can serve as a reference for evaluating other existing multistage networks. The topology of the baseline network can be generated in a recursive way. Fig. 3.2 shows the first iteration of the recursive process in which the first stage contains one $N \times N$ block and the second stage contains two $(N/2) \times (N/2)$ subblocks, C_0 and C_1. The process can recursively be applied to the subblocks in each iteration until the $N/2$ subblocks of size 2×2 are reached. To complete the process $\log_2 N-1$ iterations are needed. There are $\log_2 N$ stages of the switching elements and $\log_2 N+1$ levels of the links. There is a similarity between this baseline network and Batcher's bitonic sorting network [69]. It is therefore possible to design a switching element for both purposes. A network of

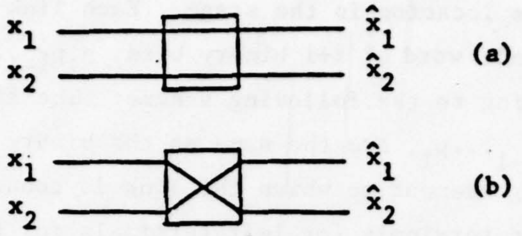


Fig. 3.1 A switching element. (a) Direct connection. (b) Crossed connection.

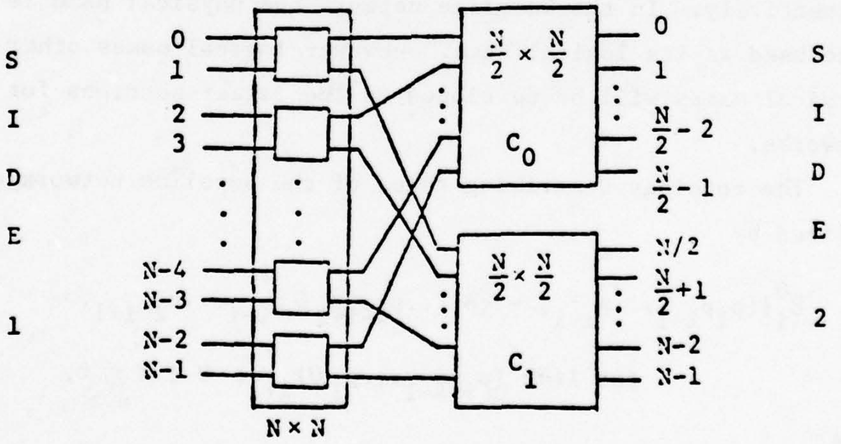


Fig. 3.2 Recursive process to construct the baseline network.

N=16 is illustrated in Fig. 3.3.

The physical names are assigned as follows. The stages are labelled in a sequence from 0 to $\log_2 N-1$ with 0 for the left most stage and $\log_2 N-1$ for the right most stage. Similarly, the levels of links are labelled in a sequence from 0 to $\log_2 N$. In each stage each switching element is named by a code word of $\ell = \log_2 N-1$ binary bits, $p_\ell p_{\ell-1} \cdots p_1$, which is the binary representation of its location in the stage. Each link in each level is named by a code word of $\ell+1$ binary bits, $p_\ell p_{\ell-1} \cdots p_0$, which is coded according to the following scheme: The first ℓ left most bits, $p_\ell p_{\ell-1} \cdots p_1$, are the same as the binary representation of the switching element to which the link is connected on one of its two right terminals (or left terminals for the case of level 0); the last bit, p_0, is equal to 0 if the link is connected to an upper terminal of the switching element and p_0 is equal to 1 if the link is connected to a lower terminal. For an example of the binary representation of the physical names of the switching elements and the links, see Fig. 3.3. Throughout the rest of this chapter the binary representations of physical names of a switching element in stage i and a link in level i are thus assigned $(p_\ell p_{\ell-1} \cdots p_1)_i$ and $(p_\ell p_{\ell-1} \cdots p_0)_i$, respectively. In the baseline network the physical name is also used as its logical name. However logical names other than physical names will be developed in the latter sections for other networks.

The topology describing rules of the baseline network are defined by

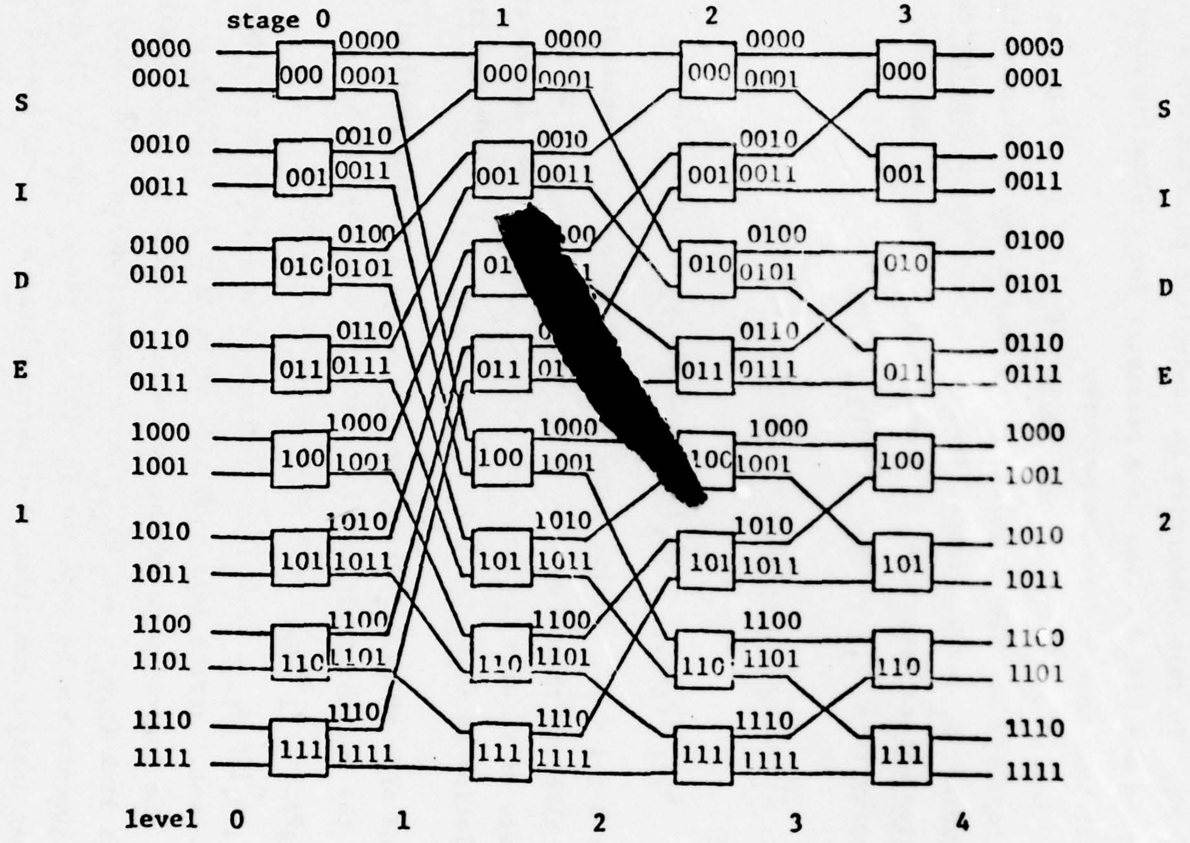
$$\beta_i^0[(p_\ell p_{\ell-1} \cdots p_1)_i] = (p_\ell \cdots p_{\ell-i+1}\, 0\, p_{\ell-i} \cdots p_2)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1} \cdots p_1\, 0)_{i+1},\ 0 \le i < \ell, \qquad (3.1)$$

and

$$\beta_i^1[(p_\ell p_{\ell-1} \cdots p_1)_i] = (p_\ell \cdots p_{\ell-i+1}\, 1\, p_{\ell-i} \cdots p_2)_{i+1}\ ,$$

$$\text{for link } (p_\ell p_{\ell-1} \cdots p_1\, 1)_{i+1},\ 0 \le i < \ell\ . \qquad (3.2)$$



[redacted]

Fig. 3.3 A baseline network with name representation.

B. Equivalent Networks

Some equivalent relationships among networks have been described in the literature [19,36,45]. However these descriptions have not formally been proven. Some analysis techniques and partial comparative study also appear in the literature [48-50]. In this subsection an equivalent relationship will be defined and will be used as a parameter for evaluating other multistage interconnection networks.

Let $\{\alpha_i^0, \alpha_i^1 \mid 0 \leq i < \ell\}$ and $\{\beta_i^0, \beta_i^1 \mid 0 \leq i < \ell\}$ be two sets of topology describing rules of two multistage interconnection networks, N_1 and N_2, respectively. Network N_1 is <u>topologically equivalent</u> to network N_2 if there is a one-to-one and onto mapping rule, γ_i, on $(p_\ell p_{\ell-1} \cdots p_1)_i$ such that

$$\beta_i^0 \gamma_i [(p_\ell p_{\ell-1} \cdots p_1)_i] = \gamma_{i+1} \alpha_i^0 [(p_\ell p_{\ell-1} \cdots p_1)_i] \ , \qquad (3.3)$$

and

$$\beta_i^1 \gamma_i [(p_\ell p_{\ell-1} \cdots p_1)_i] = \gamma_{i+1} \alpha_i^1 [(p_\ell p_{\ell-1} \cdots p_1)_i] \ , \qquad (3.4)$$

for $0 \leq i < \ell$. If two networks are topologically equivalent, we also say that they show an <u>isomorphic topology</u>. We use the image of γ_i as a logical name, $(b_\ell b_{\ell-1} \cdots b_1)_i$, and denote the relation as $(b_\ell b_{\ell-1} \cdots b_1)_i = \gamma_i [(p_\ell p_{\ell-1} \cdots p_1)_i]$. On the right hand side of Eq. (3.3), $\alpha_i^0 [(p_\ell p_{\ell-1} \cdots p_1)_i]$ is the physical name of the switching element in stage i+1 which has link $(p_\ell p_{\ell-1} \cdots p_1 0)_{i+1}$ as a communication path and $\gamma_{i+1} \alpha_i^0 [(p_\ell p_{\ell-1} \cdots p_1 0_i]$ is the logical name of this switching element. On the left hand side of Eq. (3.3), $\gamma_i [(p_\ell p_{\ell-1} \cdots p_1)_i]$ is the logical name of the switching element in stage i which has link $(p_\ell p_{\ell-1} \cdots p_1 0)_{i+1}$ as a communication path. Similar arguments can be made for Eq. (3.4). Hence, Eqs. (3.3) and (3.4) just imply a condition that there exists a logical name scheme for network N_1 such that the connectivities of the interconnection paths labelled with logical names in network N_1 can be described by the topology describing rules of network N_2.

The following theorems on the isomorphic topology are proven by showing that Eq. (3.3) and Eq. (3.4) hold. In each proof we also show an example network labelled with logical names.

Theorem 3.1: The regular SW banyan network with S=F=2 or the indirect binary n-cube network defined by

$$\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_\ell\cdots p_{i+2}\,0\,p_i\cdots p_1)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\,0)_{i+1},\ 0 \le i < \ell, \qquad (3.5)$$

and

$$\alpha_i^1[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_\ell\cdots p_{i+2}\,1\,p_i\cdots p_1)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\,1)_{i+1},\ 0 \le i < \ell, \qquad (3.6)$$

is topologically equivalent to the baseline network.

Proof: The following mapping rule, γ_i, is used to show the equivalence,

$$\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_1\cdots p_i\ p_\ell\cdots p_{i+1})_i,$$

$$\text{for } 0 \le i < \ell. \qquad (3.7)$$

It can be seen that the mapping rule provides a one-to-one and onto relationship between the physical and logical names. The logical name assignment, $(b_\ell b_{\ell-1}\cdots b_1)_i = \gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i$ on a regular SW banyan network with S=F=2 or an indirect binary n-cube network is shown in Fig. 3.4.

From Eqs. (3.1), (3.5) and (3.7) we have

$$\gamma_{i+1}\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = \gamma_{i+1}[(p_\ell\cdots p_{i+2}\,0\,p_i\cdots p_1)_{i+1}]$$

$$= (p_1\cdots p_i\,0\,p_\ell\cdots p_{i+2})_{i+1}, \qquad (3.8)$$

and

$$\beta_i^0\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = \beta_i^0[(p_1\cdots p_i\ p_\ell\cdots p_{i+1})_i]$$

$$= (p_1\cdots p_i\,0\,p_\ell\cdots p_{i+2})_{i+1}\ , \qquad (3.9)$$

for $0 \le i < \ell$. By Eqs. (3.8) and (3.9) we show that Eq. (3.3) holds. Similarly, using Eqs. (3.2), (3.6) and (3.7) we can also show that Eq. (3.4) holds. Q.E.D.

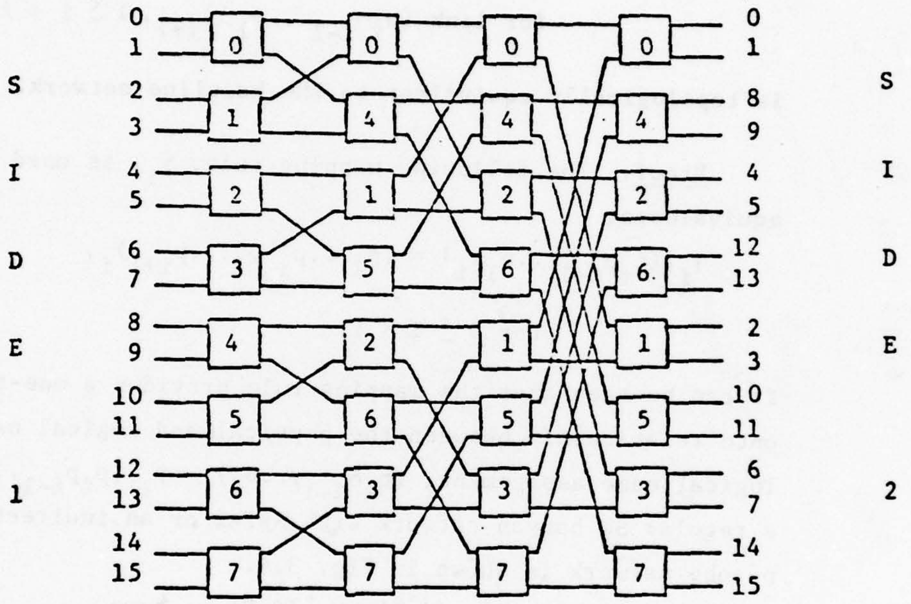


Fig. 3.4 A banyan (S=F=2), or indirect binary n-cube network structure with new configuration.

<u>Theorem 3.2</u>: The modified data manipulator defined by

$$\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_\ell\cdots p_{\ell-i+1}\,0\,p_{\ell-i-1}\cdots p_1)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\,0)_{i+1},\quad 0 \le i < \ell, \qquad (3.10)$$

and

$$\alpha_i^1[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_\ell\cdots p_{\ell-i+1}\,1\,p_{\ell-i-1}\cdots p_1)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\,1)_{i+1},\quad 0 \le i < \ell, \qquad (3.11)$$

is topologically equivalent to the baseline network.

<u>Proof</u>: The following mapping rule, γ_i, is used to show the equivalence,

$$\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_\ell\cdots p_{\ell-i+1}\,p_1\cdots p_{\ell-i})_i,$$

$$\text{for } 0 \le i \le \ell. \qquad (3.12)$$

It can be seen that the mapping rule is one-to-one and onto. The logical name assignment on a modified data manipulator of N=16 is shown in Fig. 3.5.

From Eqs. (3.1), (3.10) and (3.12) we have

$$\gamma_{i+1}\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = \gamma_{i+1}[(p_\ell\cdots p_{\ell-i+1}\,0\,p_{\ell-i-1}\cdots p_1)_{i+1}]$$

$$= (p_\ell\cdots p_{\ell-i+1}\,0\,p_1\cdots p_{\ell-i-1})_{i+1}, \qquad (3.13)$$

and

$$\beta_i^0\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = \beta_i^0[(p_\ell\cdots p_{\ell-i+1}\,p_1\cdots p_{\ell-i})_i]$$

$$= (p_\ell\cdots p_{\ell-i+1}\,0\,p_1\cdots p_{\ell-i-1})_{i+1}, \qquad (3.14)$$

for $0 \le i < \ell$. By Eqs. (3.13) and (3.14) we show that Eq. (3.3) holds. Similarly, using Eqs. (3.2), (3.11) and (3.12), we can also show that Eq. (3.4) holds. Q.E.D.

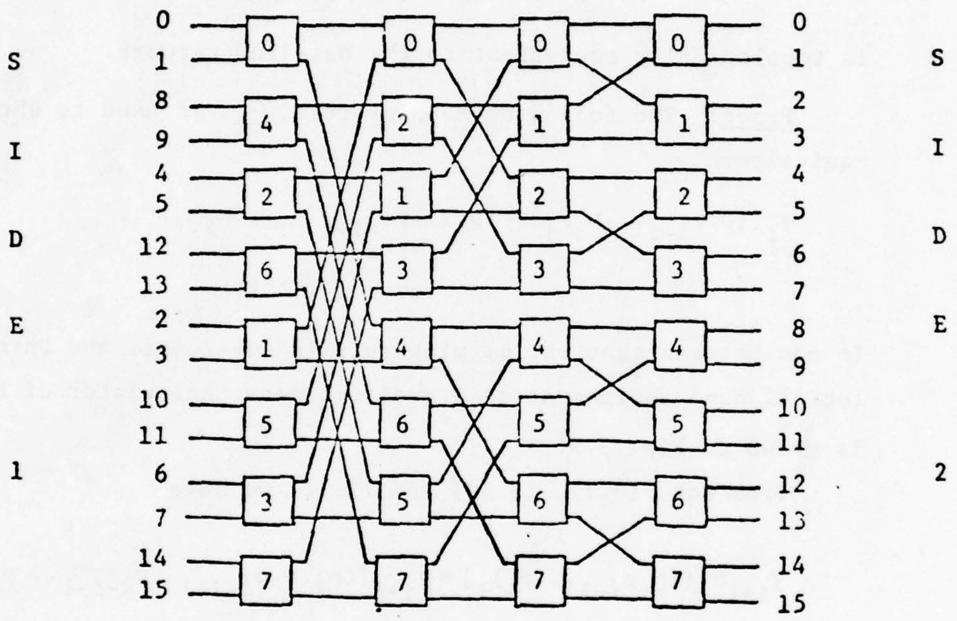


Fig. 3.5 A modified data manipulator with configuration.

<u>Theorem 3.3</u>: The flip network defined by

$$\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = (0\ p_\ell p_{\ell-1}\cdots p_2)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\ 0)_{i+1},\ 0 \leq i < \ell, \tag{3.15}$$

and

$$\alpha_i^1[(p_\ell p_{\ell-1}\cdots p_1)_i] = (1\ p_\ell p_{\ell-1}\cdots p_2)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\ 1)_{i+1},\ 0 \leq i < \ell, \tag{3.16}$$

is topologically equivalent to the baseline network.

<u>Proof</u>: The following mapping rule, γ_i, is used to show the equivalence,

$$\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_{\ell-i+1}\cdots p_\ell p_{\ell-i}\cdots p_1)_i,$$

$$\text{for } 0 \leq i \leq \ell. \tag{3.17}$$

It can be seen that the mapping rule is one-to-one and onto. The logical name assignment on a flip network of N = 16 is shown in Fig. 3.6. From Eqs. (3.1), (3.15) and (3.17), we have

$$\begin{aligned}\gamma_{i+1}\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] &= \gamma_{i+1}[(0\ p_\ell p_{\ell-1}\cdots p_2)_{i+1}]\\ &= (p_{\ell-i+1}\cdots p_\ell\ 0\ p_{\ell-i}\cdots p_2)_{i+1},\end{aligned} \tag{3.18}$$

and

$$\begin{aligned}\beta_i^0\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] &= \beta_i^0[(p_{\ell-i+1}\cdots p_\ell p_{\ell-i}\cdots p_1)_i]\\ &= (p_{\ell-i+1}\cdots p_\ell\ 0\ p_{\ell-i}\cdots p_2)_{i+1},\end{aligned} \tag{3.19}$$

for $0 \leq i < \ell$. By Eqs. (3.18) and (3.19) we show that Eq. (3.3) holds. Similarly, using Eqs. (3.2), (3.16) and (3.17) we can also show that Eq. (3.4) holds. Q.E.D.

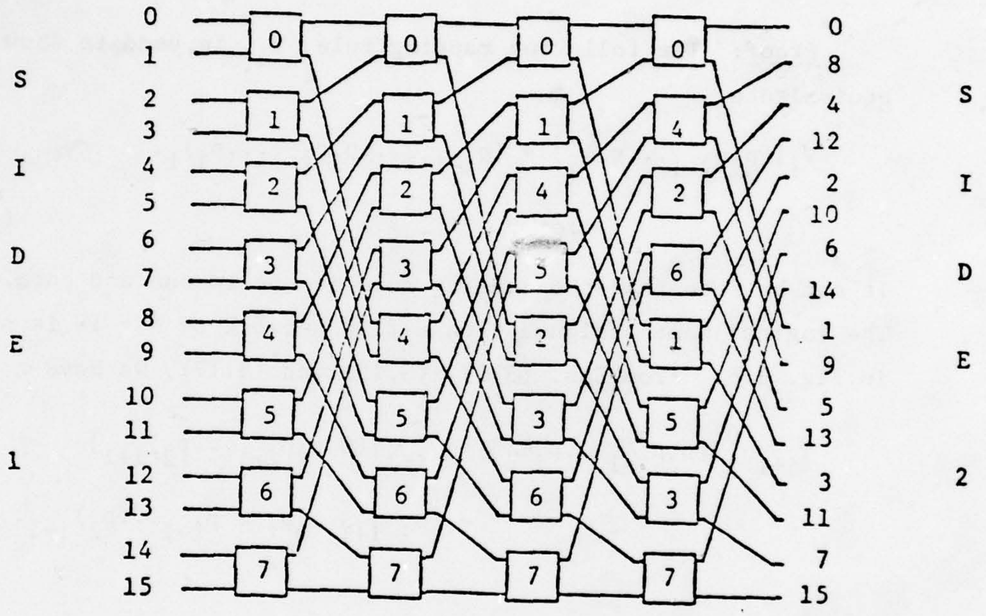


Fig. 3.6 A flip network structure with new configuration.

Theorem 3.4: The omega network defined by

$$\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_{\ell-1}p_{\ell-2}\cdots p_1\,0)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\,0)_{i+1},\ 0 \leq i < \ell, \qquad (3.20)$$

and

$$\alpha_i^1[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_{\ell-1}p_{\ell-2}\cdots p_1\,1)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\,1)_{i+1},\ 0 \leq i < \ell, \qquad (3.21)$$

is topologically equivalent to the baseline network.

Proof: The following mapping rule, γ_i, is used to show the equivalence,

$$\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_i\cdots p_1\ p_{i+1}\cdots p_\ell)_i$$

$$\text{for } 0 \leq i \leq \ell. \qquad (3.22)$$

It can be seen that the mapping rule is one-to-one and onto. The' logical name assignment on an omega network of N=16 is shown in Fig. 3.7.

From Eqs. (3.1), (3.20) and (3.22), we have

$$\gamma_{i+1}\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = \gamma_{i+1}[(p_{\ell-1}p_{\ell-2}\cdots p_1\,0)_{i+1}]$$

$$= (p_i\cdots p_1\,0\,p_{i+1}\cdots p_{\ell-1})_{i+1}, \qquad (3.23)$$

and

$$\beta_i^0\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = \beta_i^0[(p_i\cdots p_1\ p_{i+1}\cdots p_\ell)_i]$$

$$= (p_i\cdots p_1\,0\,p_{i+1}\cdots p_{\ell-1})_{i+1}, \qquad (3.24)$$

for $0 \leq i < \ell$. By Eqs. (3.23) and (3.24) we show that Eq. (3.3) holds. Similarly, using Eqs. (3.2), (3.21) and (3.22) we can also show that Eq. (3.4) holds. Q.E.D.

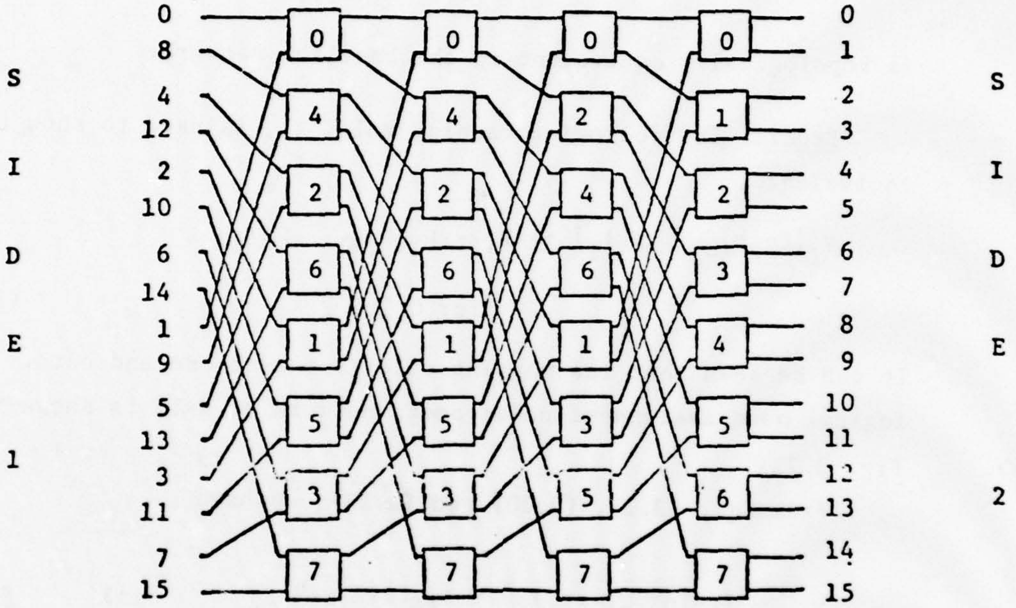


Fig. 3.7 An omega network structure with new configuration.

Theorem 3.5: The reverse baseline network defined by

$$\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_\ell \cdots p_{i+2}\ p_i \cdots p_1\ 0)_{i+1}\ ,$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\ 0)_{i+1},\ 0 \le i < \ell, \qquad (3.25)$$

and

$$\alpha_i^1[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_\ell \cdots p_{i+2}\ p_i \cdots p_1\ 1)_{i+1},$$

$$\text{for link } (p_\ell p_{\ell-1}\cdots p_1\ 1)_{i+1},\ 0 \le i < \ell, \qquad (3.26)$$

is topologically equivalent to the baseline network.

Proof: The following mapping rule, γ_i, is used to show the equivalence,

$$\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = (p_i \cdots p_1\ p_\ell \cdots p_{i+1})_i,$$

$$\text{for } 0 \le i \le \ell. \qquad (3.27)$$

It can be seen that the mapping rule is one-to-one and onto. The logical name assignment on a reverse baseline network of $N = 16$ is shown in Fig. 3.8.

From Eqs. (3.1), (3.25) and (3.27) we have

$$\gamma_{i+1}\alpha_i^0[(p_\ell p_{\ell-1}\cdots p_{\ell-1})_i] = \gamma_{i+1}[(p_\ell \cdots p_{i+2}\ p_i \cdots p_1\ 0)_{i+1}]$$

$$= (p_i \cdots p_1\ 0\ p_\ell \cdots p_{i+2})_{i+1}, \qquad (3.28)$$

and

$$\beta_i^0\gamma_i[(p_\ell p_{\ell-1}\cdots p_1)_i] = \beta_i^0[(p_i \cdots p_1\ p_\ell \cdots p_{i+1})_i]$$

$$= (p_i \cdots p_1\ 0\ p_\ell \cdots p_{i+2})_{i+1}, \qquad (3.29)$$

for $0 \le i < \ell$. By Eqs. (3.28) and (3.29) we show that Eq. (3.3) holds. Similarly, using Eqs. (3.2), (3.26) and (3.27) we can also show that Eq. (3.4) holds. Q.E.D.

The networks described in Theorems 3.1-3.5 and any other similar networks form a topologically equivalent class of multistage interconnection networks. The proofs of Theorems 3.1-3.5 also provide rules, Eq. (3.7), (3.12), (3.17), (3.22), and (3.27)

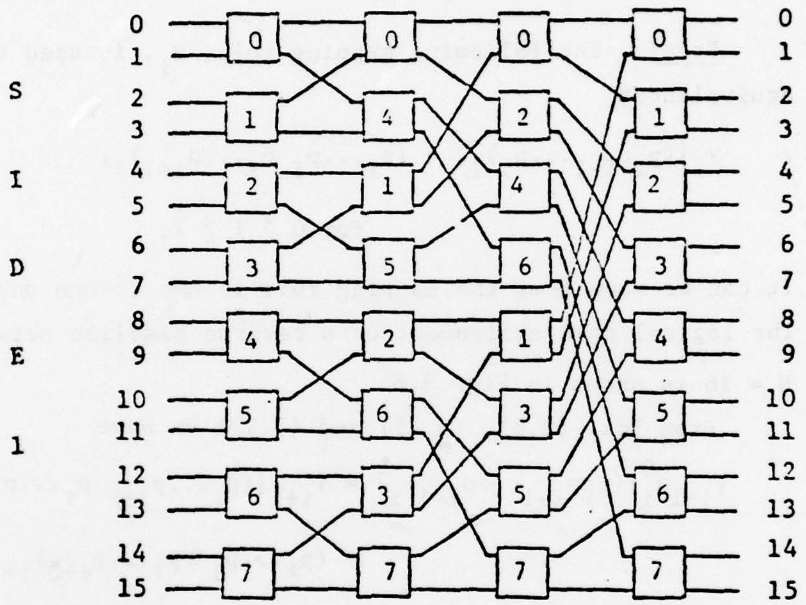


Fig. 3.8 A reverse baseline network with configuration.

for the logical name assignment on the switching elements. The logical name of each link in the network can be obtained from the logical name of the adjacent switching element according to the rules set up in Subsection 3.1.A.

Corollary 3.1: If network N_1 is shown to be topologically equivalent to network N_2 by using the one-to-one and onto mapping rule, γ_i, (i.e., $N_1 \xrightarrow{\gamma_i} N_2$), then $N_2 \xrightarrow{\gamma_i^{-1}} N_1$.

The fact of Corollary 1 is obvious since γ_i is one-to-one and onto and hence γ_i^{-1} exists.

Corollary 3.2: If network N_1 is topologically equivalent to the baseline network and the image of its mapping rule is used as the logical name, we can obtain the baseline network structure from that of network N_1 by rearranging the switching elements in each stage in the ascending order of the logical name.

The corollary follows from the fact that the interconnection paths between switching elements labelled with the logical name in network N_1 can be described by the topology describing rules of the baseline network. Thus we can compute the routing information of the baseline network from the indirect binary n-cube network, the regular SW banyan network with S=F=2, the modified data manipulator, the flip network or the omega network.

Corollary 3.3: The routing information of the indirect binary n-cube network, the regular SW banyan network with S=F=2, the modified data manipulator, the flip network and the omega network can be computed from the baseline network.

This corollary is an immediate result of Corollaries 3.1 and 3.2.

Corollary 3.4: If $N_1 \xrightarrow{\gamma_i} N_2$ and $N_2 \xrightarrow{\delta_i} N_3$ then $N_1 \xrightarrow{\delta_i \gamma_i} N_3$.

Proof: Assume the topology describing rules of N_1, N_2 and N_3 are $\{\alpha_i^0, \alpha_i^1 \mid 0 \leq i \leq \ell\}$, $\{\beta_i^0, \beta_i^1 \mid 0 \leq i \leq \ell\}$ and $\{\xi_i^0, \xi_i^1 \mid 0 \leq i \leq \ell\}$, and the logical names of switching elements in N_1 and N_2 are $(a_\ell a_{\ell-1} \ldots a_1)_i$ and $(b_\ell b_{\ell-1} \ldots b_1)_i$, respectively. By the definitions, we have

$$\beta_i^j\gamma_i[(a_\ell a_{\ell-1}\ldots a_1)_i] = \gamma_{i+1}\alpha_i^j[(a_\ell a_{\ell-1}\ldots a_1)_i] , \tag{3.30}$$

and

$$\xi_i^j\delta_i[(b_\ell b_{\ell-1}\ldots b_1)_i] = \delta_{i+1}\beta_i^j[(b_\ell b_{\ell-1}\ldots b_1)_i] , \tag{3.31}$$

for $0 \le i < \ell$ and $j = 0$ or 1.

From Eq. (3.31), we have

$$\xi_i^j\delta_i\gamma_i[(a_\ell a_{\ell-1}\ldots a_1)_i] = \delta_{i+1}\beta_i^j\gamma_i[(a_\ell a_{\ell-1}\ldots a_1)_i] , \tag{3.32}$$

for $0 \le i < \ell$ and $j = 0$ or 1.

Substituting Eq. (3.30) into Eq. (3.32) we have

$$\xi_i^j\delta_i\gamma_i[(a_\ell a_{\ell-1}\ldots a_1)_i] = \delta_{i+1}\gamma_{i+1}\alpha_i^j[(a_\ell a_{\ell-1}\ldots a_1)_i] , \tag{3.33}$$

for $0 \le i < \ell$ and $j = 0$ or 1.

Eq. (3.33) shows that N_1 and N_3 are topologically equivalent.

Q.E.D.

Using the symmetrical and transitive properties which are shown in Corollaries 3.1 and 3.4, respectively, we can describe the following corollary.

Corollary 3.5: The indirect binary n-cube network, the regular SW banyan network with S=F=2, the modified data manipulator, the flip network, the omega network, the reverse baseline network, and the baseline network are all topologically equivalent, and consequently they can share the same routing information for setting up the connection paths.

3.2 Routing Techniques

In this section we shall discuss some routing techniques for the class of the multistage interconnection networks. Overall, these techniques can be organized into a routing procedure which is called the binary tree coding method. The binary tree coding method uses a labelling scheme to facilitate a simple routing algorithm according to the concept of the reducible sets [29]. A destination tag routing method proposed for the omega network [35,36] shows an equivalent concept of the reducible sets. However, the original configuration of the omega network restricts the destination tag routing method in the one-way communications from a set of inputs to a set of outputs.

The binary tree coding method can result in the same connection path no matter which side of the terminal is chosen as input side and the other as output side so that no distinction needs to be made between the inputs and the outputs.

A. Labelling Scheme for the Terminal Link

The label assigned to a terminal link can be considered as the address of the processing unit attached to that terminal link. The labelling scheme for the terminal link has been shown in the logical name assignment in the previous section. However, we will describe a short cut scheme which can be used to label the terminal link without going into the mapping rules. A binary tree can be formed by choosing any one of the switching elements in stage 0 as the root and iteratively considering the adjacent switching elements in the next stage as the nodes of the binary tree. There are two outgoing links from the root and every node in the binary tree. The label of each terminal link on Side 2 can be obtained by assigning weight 0 for the upper outgoing link, and 1 for the lower one and concatenating the weight along the path from the root to the terminal link. There are N binary trees which can be formed for labelling purposes and each tree results in the same labels. Fig. 3.9(a) shows an example for an omega network. For labelling terminal links on Side 1, a binary tree can be similarly formed by using one of the switching elements in the right most stage as the root (see Fig. 3.9(b)).

B. Routing Algorithm

A simple algorithm follows from the binary coding of the terminal links. In the algorithm each terminal link is assigned to a binary tree using the adjacent switching element as the root. For each connection request from the source terminal link to the destination link(s), the routing algorithm sets up a subtree of the binary tree assigned to the source terminal link according to the binary representations of the destination labels. For a simple demonstration we first consider the one-to-one connection

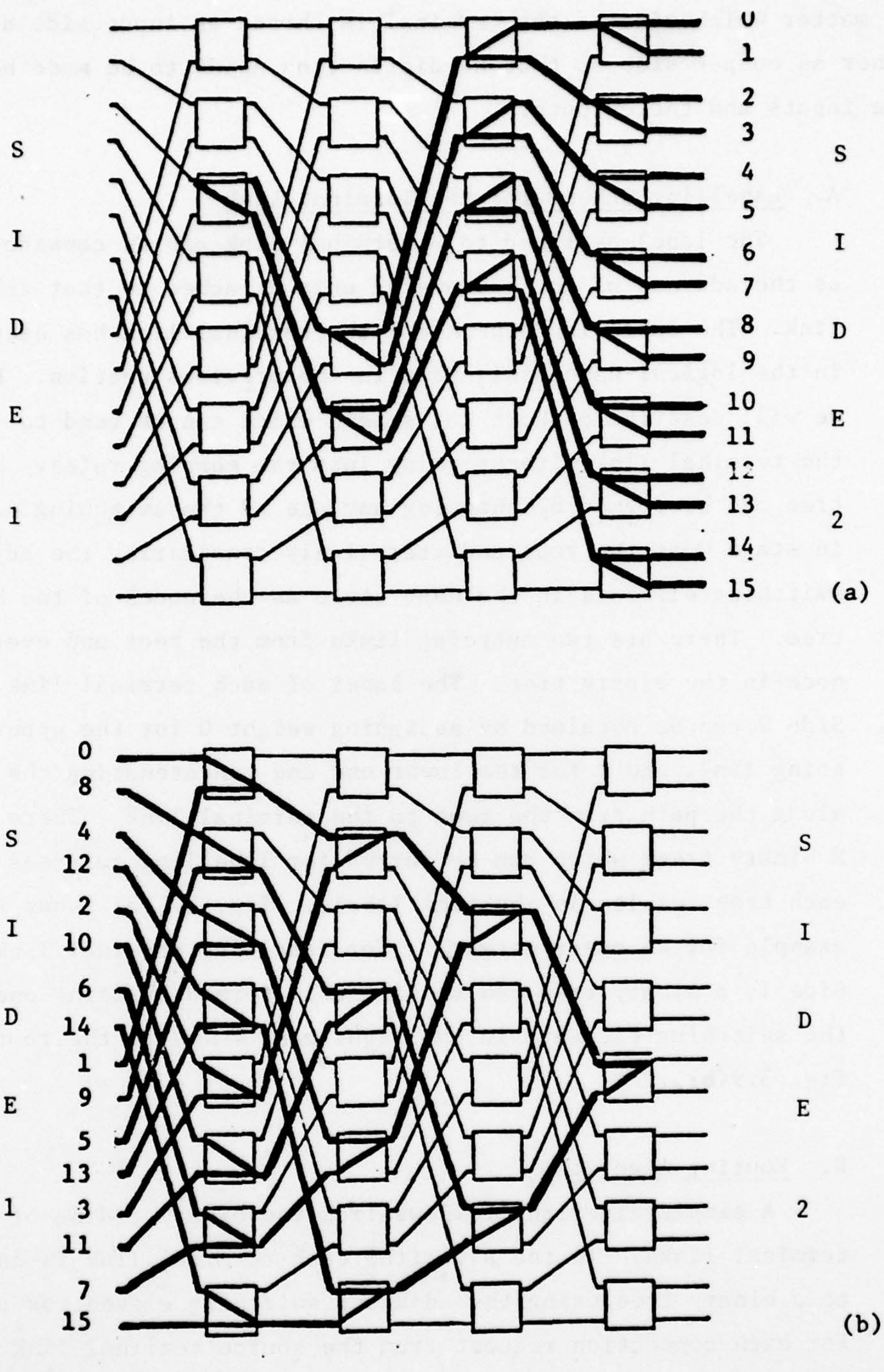


Fig. 3.9 Binary tree coding of omega network.
(a) Right side. (b) Left side.

request. Let source terminal link A labelled by $a_\ell a_{\ell-1}\cdots a_0$ on Side 1 be connected to destination terminal link Z labelled by $z_\ell z_{\ell-1}\cdots z_0$ on Side 2. Starting at A, the first node (root of the tree) to which A is connected is set to switch A to the upper link if $z_\ell = 0$ or the lower link if $z_\ell = 1$. The second node in the path is again set to switch A to the upper link if $z_{\ell-1} = 0$ or the lower link if $z_{\ell-1} = 1$. This scheme is continued until we get the proper destination. The path connected is one part of the binary tree assigned to source A. An example is shown by path 1 in Fig. 3.10. If we consider Z as the source terminal link and A as the destination terminal link, the same procedure will lead us to choose the same path. At this point, properties of the completeness and homogeneity of the routing algorithm will be proven.

Theorem 3.6: Completeness: The binary tree coding method can set up any mapping connection from one terminal on one side of terminals to another terminal of the other side. Homogeneity: The binary tree coding method will lead to the same path between these two terminals no matter which end terminal is chosen as the source terminal so that no distinction needs to be made between the inputs and the outputs.

Proof: Completeness: Since the binary representation of a destination terminal in both sides is the same as the code word formed by concatenating the weight of the link in the path from the source terminal to the destination terminal, we can see that any source terminal on one side can be connected to an arbitrary terminal on the other side by using the routing algorithm. Hence, the routing algorithm can connect the terminal pair specified in any connection request. Homogeneity: The homogeneity can be proven by showing that the two sets of the switching elements respectively in the two paths set up in opposite directions are identical. Assume again a source terminal $A = a_\ell a_{\ell-1}\cdots a_0$ on Side 1 is to be connected to a destination terminal $Z = z_\ell z_{\ell-1}\cdots z_0$ on Side 2. From Eqs. (3.1) and (3.2) we can see that the first switching element in the path from A to Z is $(a_\ell a_{\ell-1}\cdots a_1)_0$, the second one is $(z_\ell a_\ell a_{\ell-1}\cdots a_2)_1$, and the third one is $(z_\ell z_{\ell-1} a_\ell a_{\ell-1}\cdots a_3)_2$.

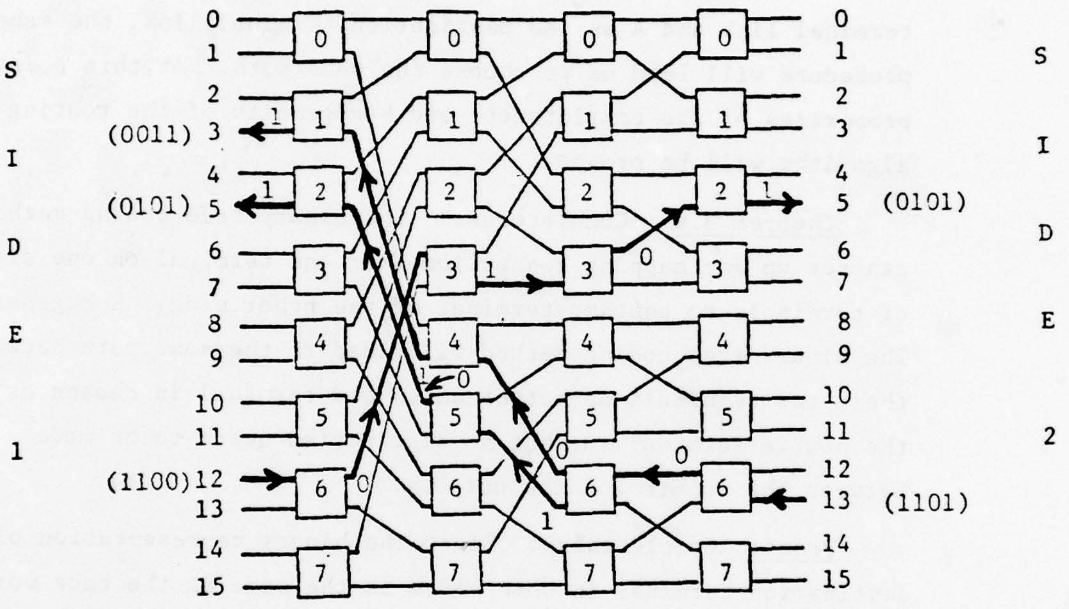


Fig. 3.10 Path routing (path 1 ⟶ ;path 2 ⟵).

In general, the set of the switching elements which are in the connected path is

$$S_1 = \{(z_\ell \ldots z_{\ell-i+1} a_\ell a_{\ell-1} \ldots a_{i+1})_i \mid 0 \le i \le \ell\} . \qquad (3.34)$$

Similarly, the topology describing rules, Eqs. (3.25) and (3.26), of the reverse baseline network can be used to compute the set of in-path switching elements if we choose Z as the source and A as the destination. Considering Z is on Side 1 of the reverse baseline network, we can see that the first switching element is $(z_\ell z_{\ell-1} \ldots z_1)_0$, the second one is $(z_\ell z_{\ell-1} \ldots z_2 a_\ell)_1$, and the third one is $(z_\ell z_{\ell-1} \ldots z_3 a_\ell a_{\ell-1})_2$. In general, the set of in-path switching elements becomes

$$S_2 = \{(z_\ell z_{\ell-1} \ldots z_{j+1} a_\ell a_{\ell-1} \ldots a_{\ell-j-1})_{\ell-j} \mid 0 \le j \le \ell\}. \qquad (3.35)$$

Since, in the baseline network, $j = \ell-i$, we have $S_1 = S_2$ by Eqs. (3.34) and (3.35). This result shows that the two sets of the switcning elements respectively in the two paths set up in opposite directions are identical. Q.E.D.

The one-to-one connection request is considered to be a special case of the one-to-many connection request. For a one-to-many connection request there are as many source-destination pairs as the number of destination terminals specified in the request. The switching element set and the link set of the subtree for a one-to-many connection request can be obtained by unioning respective sets of each individual source-destination path. An example is shown in path 2, in Fig. 3.10, in which a subtree is set up for the one-to-two connection request of source-destination pairs from terminal 13 on Side 2 to terminal 3 and terminal 5 on Side 1.

C. Conflict Resolution

Since all networks in the class of multistage interconnection networks are of blocking type, an effort of restructuring and recompiling the computation algorithms into algorithms which fully utilize the network connectivities can enhance the system performance. However, not all computation algorithms can be restructured and recompiled into fully utilizing algorithms. A conflict resolution algorithm is necessary. The sharing of a common link by two or more independent subtrees is called a conflict. The algorithm detects any conflicts and resolves the conflicts by deferring some connection requests in the given request set so that all the connection requests that remain show no conflicts. Before we can detect the conflicts, we should compute the set of links which must be connected in a subtree for a connection request. In the proof of Theorem 3.6 we have shown, in Eq. (3.34), that the in-path switching element in stage i for the mapping from $a_\ell a_{\ell-1}\ldots a_0$ on Side 1 to $z_\ell z_{\ell-1}\ldots z_0$ on Side 2 can be expressed as $(z_\ell\ldots z_{\ell-i+1}a_\ell\ldots a_{i+1})_i$. Hence the in-path link in level i+1 can be expressed as $(z_\ell\ldots z_{\ell-i+1}a_\ell\ldots a_{i+1}z_{\ell-i})_{i+1}$. We can also have another expression to count the links for the mapping from $z_\ell z_{\ell-1}\ldots z_0$ to $a_\ell a_{\ell-1}\ldots a_0$. But it turns out that the two expressions can lead to the same result. Without losing any generality, we will work on the former expression designed for the connection requests from Side 1 to Side 2. For the sake of clarity the binary representation for the links are converted into decimal numbers. For example, the ordered set of links for the mapping shown in path 1 in Fig. 3.10 is expressed as follows:

$$P = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 12 \\ 5 \end{pmatrix} = \{12,12,7,6,5\}.$$

Now we are able to detect the conflicts. We shall demonstrate the scheme by an example which was used earlier by Opferman and Tsao-Wu [29]. It contains 15 connection requests except the one from source 0:

$$P = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 11 & 15 & 4 & 2 & 6 & 1 & 7 & 5 & 8 & 9 & 12 & 14 & 3 & 13 & 10 \end{pmatrix}$$

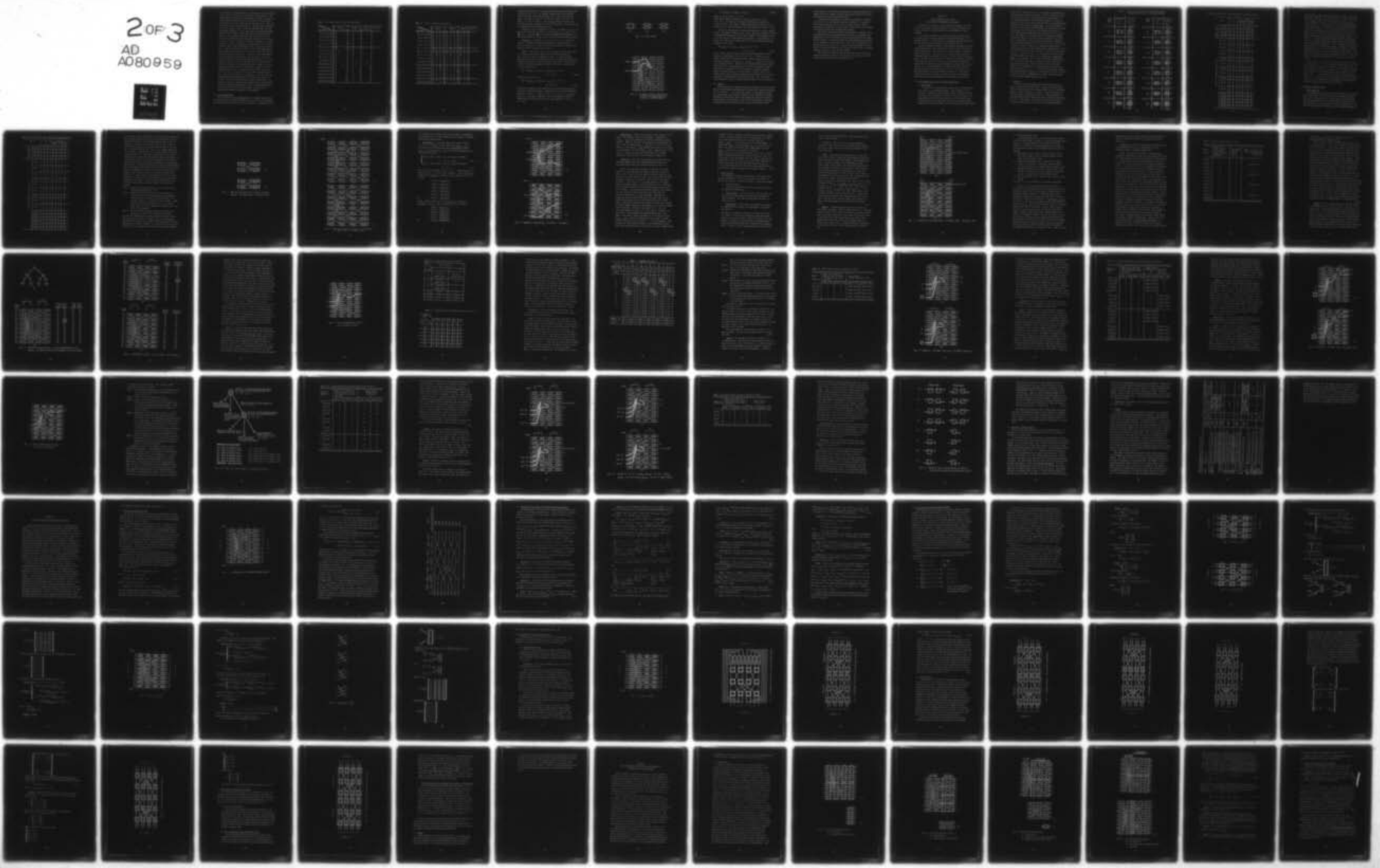
The conflicts for request set P are shown in Table 3.1. In this example, there are conflicts at some links of level 1, 2 and 3, and there is no conflict in level 4. The entries of the conflicts table are filled in this way. If an i^{th} link set (row) contains a j^{th} conflict link (column) then the i-j entry

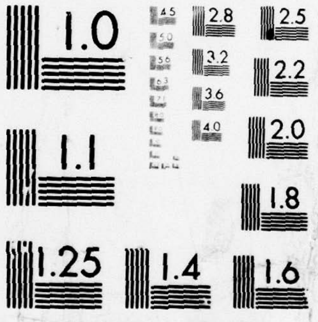
Table 3.1 A Conflict Table.

Conflict Link	Level 1				Level 2				Level 3			Level 4
Link Set	4	6	11	15	2	3	12	15	5	10	14	
{1, 1, 8, 9,11}												
{2, 3, 9,13,15}												
{3, 2, 1, 4, 4}												
{4, 4, 2, 1, 2}	X				X							
{5, 4, 3, 5, 6}	X					X			X			
{6, 6, 2, 0, 1}		X			X							
{7, 6, 3, 5, 7}		X				X			X			
{8, 8, 5, 6, 5}												
{9, 9,12,10, 8}							X			X		
{10,11,12,10, 9}			X				X			X		
{11,11,13,14,12}			X								X	
{12,13,15,15,14}								X				
{13,12, 6, 3, 3}												
{14,15,15,14,13}				X				X			X	
{15,15,14,11,10}				X								

AD-A080 959 WAYNE STATE UNIV DETROIT MICH F/G 9/2
INTERCONNECTION NETWORKS IN MULTIPLE-PROCESSOR SYSTEMS.(U)
DEC 79 T FENG, C WU F30602-76-C-0282
UNCLASSIFIED RADC-TR-79-304 NL

2 OF 3
AD
A080959





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

of the table is filled with an X. Otherwise, it is left unfilled. In the conflict table, each column has at least two X's. The conflicts resolution problem then becomes to choose a minimal or nearly minimal deferred set of rows such that after deleting those chosen rows each column has one and only one X. There are many ways to choose the deferred set. We shall show a possible way in the conflict table. First, we can weigh each link set by adding the number of X's on that link set row and the number of X's on the columns in which the link set being considered has an X. For example, there are three X's on link set row {5,4,3,5,6}, which indicates that the conflicts occur at link 4 of level 1, link 3 of level 2, and link 5 of level 3. As shown in Table 3.1, there is another X on each column of these three conflict links. Hence the weight of row {5,4,3,5,6} is equal to 6. Then we can choose the link set with the highest weight as one member of the deferred set, mark the chosen one with a √ and delete entries in that row. In Table 3.1 the link ordered set of {5,4,3,5,6}, {7,6,3,5,7}, {10,11,12,10,9}, and {14,15, 15,14,13} happen to have the same highest weight, 6. We choose {5,4,3,5,6} as the one to be deferred, mark it and delete the entries in that row. Now a reduced conflict table can be constructed by deleting all the columns with a single X. In the reduced table, Table 3.2, columns labelled with 4 in level 1, with 3 in level 2, and with 5 in level 3 have been deleted and {5,4,3,5,6} has been chosen to be deferred. The same cycle can be repeated on the reduced table until there is no conflict column. The link sets marked with √ form a deferred set. The link sets other than those in the deferred set are conflict-free and can be passed by the network. Table 3.3 shows the result of the conflict resolution for the example. There are four requests that should be deferred in the example.

3.3 Full Communication

The capability of full communication of a network is meant to be the ability of the network to be able to connect one of its terminals to other terminals at either side of the network. To achieve the full

Table 3.2 A Reduced Table of Conflict Resolution.

Conflict Link	Level 1				Level 2				Level 3			Level 4
Link Set	4	6	11	15	2	3	12	15	5	10	14	
{1, 1, 8, 9,11}												
{2, 3, 9,13,15}												
{3, 2, 1, 4, 4}												
{4, 4, 2, 1, 2}	X				X							
√{5, 4, 3, 5, 6}	X					X			X			
{6, 6, 2, 0, 1}		X			X							
{7, 6, 3, 5, 7}		X				X			X			
{8, 8, 5, 6, 5}												
{9, 9,12,10, 8}							X			X		
{10,11,12,10, 9}			X				X			X		
{11,11,13,14,12}			X								X	
{12,13,15,15,14}								X				
{13,12, 6, 3, 3}												
{14,15,15,14,13}				X				X			X	
{15,15,14,11,10}				X								

Table 3.3 Result of Conflict Resolution.

Conflict Link	Level 1				Level 2				Level 3			Level 4
Link Set	4	6	11	15	2	3	12	15	5	10	14	
{1, 1, 8, 9,11}												
{2, 3, 9,13,15}												
{3, 2, 1, 4, 4}												
{4, 4, 2, 1, 2}	X				X							
✓{5, 4, 3, 5, 6}	X					X			X			
✓{6, 6, 2, 0, 1}		X			X							
{7, 6, 3, 5, 7}		X				X			X			
{8, 8, 5, 6, 5}												
{9, 9,12,10, 8}							X			X		
✓{10,11,12,10, 9}			X				X			X		
{11,11,13,14,12}			X								X	
{12,13,15,15,14}								X				
{13,12, 6, 3, 3}												
✓{14,15,15,14,13}				X				X			X	
{15,15,14,11,10}				X								

communication capability, a three-state switch can be introduced [50] as shown in Fig. 3.11. The mapping requests made on the baseline network with the three-state switches can be classified into four types: (1) Side 1 to Side 2; (2) Side 2 to Side 1; (3) Side 1 to Side 1; (4) Side 2 to Side 2. The routing procedure for the first two types has been discussed in the previous section. In this section we only discuss the remaining two types.

Assume that the two terminals $A = a_\ell a_{\ell-1} \ldots a_0$ and $Z = z_\ell z_{\ell-1} \ldots z_0$ are on the same side, say Side 1. Define $c_\ell c_{\ell-1} \ldots c_0 = a_\ell a_{\ell-1} \ldots a_0 \oplus z_\ell z_{\ell-1} \ldots z_0$ where $\oplus$ is a bit-by-bit EXCLUSIVE OR operation. A routing procedure to connect A and Z is given by the following theorem.

Theorem 3.7: Assuming $c_i = 1$ and $c_j = 0$ for $j > i$, there are exactly 2^i possible shortest paths defined to include states of switching elements to connect A and Z and the number of links in the shortest path is exactly equal to $2(i+1)$.

Proof: Since A and Z are on the same side of the network, there is at least one switching element in the third-state shown in Fig. 3.11(c) in the path connecting A and Z. To count how many paths which can connect A and Z, we can count the number of switching elements which can be in the third-state in the path. From Eq. (3.34) the set of switching elements in stage k which can be reached from A can be expressed as:

$$S_A^k = \{(d_{k-1} d_{k-2} \ldots d_0 \; a_\ell a_{\ell-1} \ldots a_{k+1})_k \mid d_j = 0 \text{ or } 1; \quad 0 \leq j \leq k-1\}. \tag{3.36}$$

Similarly the set for Z is

$$S_Z^k = \{(d_{k-1} d_{k-2} \ldots d_0 \; z_\ell z_{\ell-1} \ldots z_{k+1})_k \mid d_j = 0 \text{ or } 1; \quad 0 \leq j \leq k-1\}. \tag{3.37}$$

The shortest connection path should be the one in which the only one third-state switching element, T, should be the one in S_A^k and S_Z^k with maximum k. Since $c_i = 1$ and $c_j = 0$ for $j > i$, $a_\ell = z_\ell, \ldots, a_{i+1} = z_{i+1}$ and $a_i \neq z_i$. By Eqs. (3.36) and (3.37), the maximum k to make a common element in S_A^k and S_Z^k is the one which makes $k+1 = i+1$, i.e., $k=i$. Hence

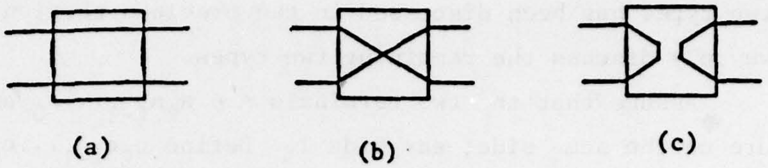


Fig. 3.11 A full switch.

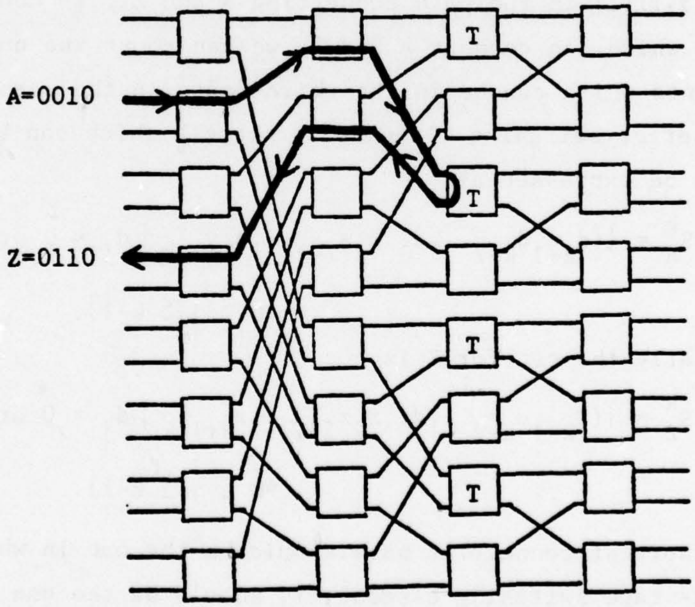


Fig. 3.12 An example of full communication. (T is the third-state switching element in a possible path.)

$$T = (d_{i-1}d_{i-2}\ldots d_0 \; a_\ell a_{\ell-1}\ldots a_{i+1})_i \; , \qquad (3.38)$$

where $d_j = 0$ or 1 and $0 \le j \le i-1$.

There are 2^i possible values for T since d_j can be 0 or 1 for $0 \le j \le i-1$. So there are 2^i possible shortest paths which can connect A and Z. The number of links which connect T and A or Z is equal to i+1 so that the length of the shortest path is 2(i+1). Q.E.D.

Theorem 3.7 demonstrates a routing procedure for the full communication. An example is shown in Fig. 3.12. Firstly, we compute i using an EXCLUSIVE OR operation on the logical names of two terminals, A and Z, on the same side, which should be connected. In the example, i is equal to 2. Next we can compute the set of the third-state switching elements in the paths:

$$\{(d_{i-1}d_{i-2}\ldots d_0 \; a_\ell a_{\ell-1}\ldots a_{i+1})_i \mid d_j = 0 \text{ or } 1; \; 0 \le j \le i-1\}. \qquad (3.39)$$

By Eq. (3.39), the set of the third-state switching element in the example is $\{(d_1d_00)_2 \mid d_0 = 0 \text{ or } 1; \; d_1 = 0 \text{ or } 1\}$. Thirdly, we can compute some of the 2^i possible shortest paths by using A and Z as the source terminals and $(d_{i-1}d_{i-2}\ldots d_0 \; a_\ell a_{\ell-1}\ldots a_{i+1})_i$ as the destination switching element. However, in the implementation phase a scheme should be set up to determine which one in the 2^i possible shortest paths should be computed. There are totally 2^2 possible shortest paths for connecting A and Z in the example. Finally, one of the possible paths can be chosen by applying the procedure for the conflict resolution as described in the previous section. Fig. 3.12 shows a connecting path in which switching element $(010)_2$ is in the third-state.

3.4 Summary

We have presented a baseline network to evaluate the relationships among the multistages interconnection networks which have been proposed in the literature. It is shown that a class of topologically equivalent multistages interconnection networks can be obtained by properly permuting the switching elements and associated links of the baseline network within the same stage. The class of topologically equivalent multistages interconnection networks includes the indirect binary n-cube network, the modified data manipulator, the flip network, the omega network, the regular SW banyan network with S=F=2, the reverse baseline network and the baseline network.

A logical name representation scheme is developed to configure this class of the topologically equivalent networks. It has been shown that one network in this class can share the same routing information developed for another network in this same class if these two networks use the same representation scheme.

The logical name representation scheme enables a simple routing algorithm and the routing algorithms are proven to be complete and homogeneous so that no distinction should be made between the inputs and the outputs. A routing procedure has been developed on the basis of the homogeneous routing algorithm. Since all the networks in the defined class are blocking, the routing procedure includes the capability to resolve the conflicts by choosing a deferred set of mapping requests according to some priority scheme.

The routing procedure can also be extended to allow one-to-one connections between all pairs of terminals so that there is no need to divide the terminals into two disjoint sets.

CHAPTER 4

FAULT-DIAGNOSIS FOR A CLASS OF MULTISTAGE INTERCONNECTION NETWORKS

The reliable operations of interconnection networks are important to the overall system performance. Yet there have been very little activities in investigating the fault-diagnosos of such networks.

It is shown, in Chapter 3, that the routing procedure of the multistage interconnection networks can be developed based on the baseline network. In this chapter we investigate the fault-detection and the fault-location problems of this baseline network.

The fault-diagnosis problem is approached by generating suitable fault-detection and fault-location test sets for every fault in the assumed fault model. The test sets are then trimmed to a minimum or nearly minimal sets. In Section 4.1 we propose a fault model of a switching element and derive a test set for every fault in the fault model. The objective of Sections 4.2 and 4.3 is to develop a specific fault-diagnosis scheme for the network constructed of switching elements having direct- and crossed-connection capabilities as shown in Fig. 3.1. The fault-diagnosis of single faults and the fault characteristics are discussed in Section 4.2. The multiple fault detection problem is then considered in Section 4.3. The fault characteristics which provide a specific criterion for designing easily testable switching elements are summarized in Section 4.4.

4.1 Fault Model and Test Set of a Switching Element

A. Fault Model

A fault in an interconnection network can be located either at a link or in a switching element. All discussion in this chapter is confined to solid logical faults. The fault located at a link can be considered to be one of line stuck types, i.e., stuck-at-zero (s-a-0) or stuck-at-one (s-a-1). We use a functional approach to consider fault types in a switching element. Generally, a switching element with two input lines and two output lines can be considered as a 2 × 2 crosspoint switching matrix which may have as many as 16 states. Table 4.1 shows the set S of the 16 states and the related symbolic representation. In our proposed multistage interconnection network, a switching element can only be in some of the 16 states by an implementation. We denote these states as valid states. In the flip network and the indirect binary n-cube network, the valid states include S_5 and S_{10}. The valid states in the omega network and the regular SW banyan network with F=S=2 include S_3, S_5, S_{10}, and S_{12}. The number of valid states which a switching can assume in order to achieve the network function depends on the capability requirement of the interconnection network. However, a faulty switching element can be in any one of the 16 states from a given valid state. Hence, for a switching element with n valid state, there are $(16)^n$ possible state combinations in which the faulty switching element can behave. We use the ordered set $\{(s_1,s_2,\ldots,s_n) \mid s_i \in S, 1 \leq i \leq n\}$ to describe the state combinations and name each of the state combinations a functional state. As an example, Fig. 3.1 shows a switching element with two valid states, S_{10} and S_5. Assume the first valid state is S_{10} and the second S_5. The functional states of switching element can be expressed by a functional state set which is an inner product set of S, S × S. There are 256 functional states. The state combination (S_{10},S_5) is the normal functional state and the other 255 state combinations are faulty functional states of the switching element shown in Fig. 3.1.

B. Test Set

A test set should be developed for detecting every fault in the fault model described above. Faults to be detected and tests for detecting them are listed in Table 4.2 for a switching element in valid state S_{10}. In Table 4.2 the detection of the link stuck fault is described in Part I. The superscripts of the link labels indicate whether the fault causes the link stuck at 0 or 1. For example, in the first row, we can see that if we apply an input $(x_1,x_2) = (1,0)$ to the switching element in

Table 4.1 Set of the 16 States and the Related Symbolic Representation of a 2 × 2 Switching Element.

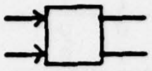
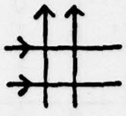
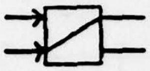
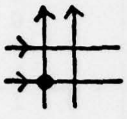
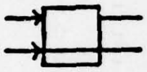
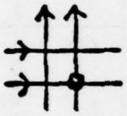
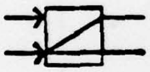
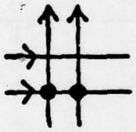
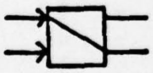
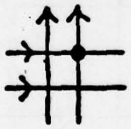
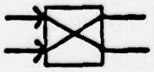
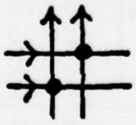
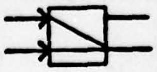
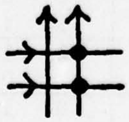
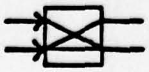
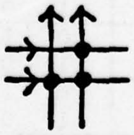
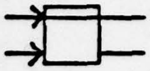
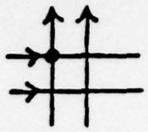
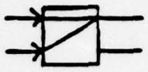
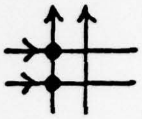
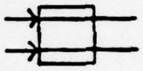
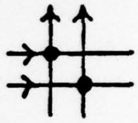
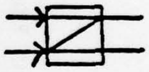
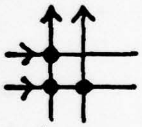
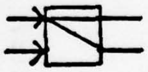
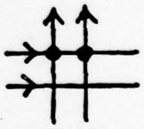
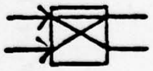
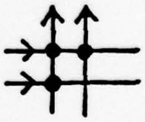
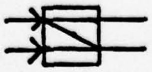
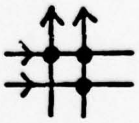
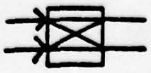
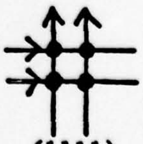
State Name	Switching Box Symbol	Crosspoint Switching Matrix Symbol
S_0		 (0000)
S_1		 (0001)
S_2		 (0010)
S_3		 (0011)
S_4		 (0100)
S_5		 (0101)
S_6		 (0110)
S_7		 (0111)
S_8		 (1000)
S_9		 (1001)
S_{10}		 (1010)
S_{11}		 (1011)
S_{12}		 (1100)
S_{13}		 (1101)
S_{14}		 (1110)
S_{15}		 (1111)

Table 4.2 Faults, Test Inputs and Outputs in Valid State S_{10}.

Fault		Test		Output			
				Normal		Faulty	
		x_1	x_2	$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_1$	$\hat{x}_2$
Part I: Link Stuck Fault	$x_1^0, \hat{x}_1^0$	1	0	1	0	0	0
		1	1	1	1	0	1
	$x_1^1, \hat{x}_1^1$	0	0	0	0	1	0
		0	1	0	1	1	1
	$x_2^0, \hat{x}_2^0$	0	1	0	1	0	0
		1	1	1	1	1	0
	$x_2^1, \hat{x}_2^1$	0	0	0	0	0	1
		1	0	1	0	1	1
Part II: Switching Element Fault	S_{10}-S_0	0	1	0	1	-	-
		1	0	1	0	-	-
		0	0	0	0	-	-
		1	1	1	1	-	-
	S_{10}-S_1	0	1	0	1	1	-
		1	0	1	0	0	-
		0	0	0	0	0	-
		1	1	1	1	1	-
	S_{10}-S_2	0	1	0	1	-	1
		1	0	1	0	-	0
		0	0	0	0	-	0
		1	1	1	1	-	1
	S_{10}-S_3	0	1	0	1	1	1
		1	0	1	0	0	0
	S_{10}-S_4	0	1	0	1	-	0
		1	0	1	0	-	1
		0	0	0	0	-	0
		1	1	1	1	-	1
	S_{10}-S_5	0	1	0	1	1	0
		1	0	1	0	0	1
	S_{10}-S_6	0	1	0	1	-	ϕ
		1	0	1	0	-	ϕ
		0	0	0	0	-	0
		1	1	1	1	-	1
	S_{10}-S_7	0	1	0	1	1	ϕ
		1	0	1	0	0	ϕ
	S_{10}-S_8	0	1	0	1	0	-
		1	0	1	0	1	-
		0	0	0	0	0	-
		1	1	1	1	1	-
	S_{10}-S_9	0	1	0	1	ϕ	-
		1	0	1	0	ϕ	-
		0	0	0	0	0	-
		1	1	1	1	1	-
	S_{10}-S_{11}	0	1	0	1	ϕ	1
		1	0	1	0	ϕ	0
	S_{10}-S_{12}	0	1	0	1	0	0
		1	0	1	0	1	1
	S_{10}-S_{13}	0	1	0	1	ϕ	0
		1	0	1	0	ϕ	1
	S_{10}-S_{14}	0	1	0	1	0	ϕ
		1	0	1	0	1	ϕ
	S_{10}-S_{15}	0	1	0	1	ϕ	ϕ
		1	0	1	0	ϕ	ϕ

valid state S_{10}, the normal output will be $(\hat{x}_1,\hat{x}_2) = (1,0)$ and the fault, x_1° or $\hat{x}_1^\circ$, of x_1 or $\hat{x}_1$ sticking at 0 will cause output to be $(\hat{x}_1,\hat{x}_2) = (0,0)$. We then say that the input $(x_1,x_2) = (1,0)$ can detect the fault, x_1° or $\hat{x}_1^\circ$, of the switching element in valid state S_{10}. The detection of switching element faults is described in Part II. For a switching element stuck in S_5, S_{10}-S_5, (see row S_{10}-S_5 of Part II of Table 4.2), if we apply input $(x_1,x_2) = (0,1)$, the faulty output will be $(\hat{x}_1,\hat{x}_2) = (1,0)$ which is different from the normal output $(\hat{x}_1,\hat{x}_2) = (0,1)$. In Table 4.2, "-" means logically undefined output and "ϕ" means logically erroneous output where 0 and 1 are the simultaneous inputs. The output values of "-" and "ϕ" depend on the circuit implementation of the switching element. However, an arbitrary assignment of 0 or 1 to "-" or "ϕ" would not affect the differentiation between the normal output and the faulty output. Hence whether we can easily design an equipment to detect "-" and "ϕ" would not disturb our development of the test set for various faults.

From Table 4.2 it can be seen that only two tests $(x_1,x_2) = (0,1)$ and $(x_1,x_2) = (1,0)$, are needed to detect all faults. The test vectors on x_1 and x_2 are 01 and 10, respectively. For an easy reference we define $t = 01$ and $\bar{t} = 10$. The same conclusion can be drawn for a switching element in valid state S_5. The faults, the test inputs and the test outputs of S_5 are shown in Table 4.3. Similarly, the above techniques can also be extended to detect faulty elements in other networks such as omega and banyan with additional valid states S_3 and S_{12}.

4.2 Diagnosis of Single Faults

A. Fault Detection

An algorithm for deriving efficient test sets for the network will be presented. The basic idea is to establish connection paths and to label each link in a path with t (or $\bar{t}$) such that each switching element in the network has its two input lines labelled with the two test vectors (t and $\bar{t}$), respectively. The connection paths are established by putting switching elements

Table 4.3 Faults, Test Inputs and Outputs in Valid State S_5.

Fault		Test		Output			
				Normal		Faulty	
		x_1	x_2	$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_1$	$\hat{x}_2$
Part I: Link Stuck Fault	x_1°, x_2°	1	0	0	1	0	0
		1	1	1	1	1	0
	$x_1^1, \hat{x}_2^1$	0	0	0	0	0	1
		0	1	1	0	1	1
	$x_2^{\circ}, \hat{x}_1^{\circ}$	0	1	1	0	0	0
		1	1	1	1	0	1
	$x_2^1, \hat{x}_1^1$	0	0	0	0	1	0
		1	0	0	1	1	1
Part II: Switching Element Fault	S_5-S_0	0	1	1	0	-	-
		1	0	0	1	-	-
		0	0	0	0	-	-
		1	1	1	1	-	-
	S_5-S_1	0	1	1	0	1	-
		1	0	0	1	0	-
		0	0	0	0	0	-
		1	1	1	1	1	-
	S_5-S_2	0	1	1	0	-	1
		1	0	0	1	-	0
		0	0	0	0	-	0
		1	1	1	1	-	1
	S_5-S_3	0	1	1	0	1	1
		1	0	0	1	0	0
	S_5-S_4	0	1	1	0	-	0
		1	0	0	1	-	1
		0	0	0	0	-	0
		1	1	1	1	-	1
	S_5-S_6	0	1	1	0	-	ϕ
		1	0	0	1	-	ϕ
		0	0	0	0	-	0
		1	1	1	1	-	1
	S_5-S_7	0	1	1	0	1	ϕ
		1	0	0	1	0	ϕ
	S_5-S_8	0	1	1	0	0	-
		1	0	0	1	1	-
		0	0	0	0	0	-
		1	1	1	1	1	-
	S_5-S_9	0	1	1	0	ϕ	-
		1	0	0	1	ϕ	-
		0	0	0	0	0	-
		1	1	1	1	1	-
	S_5-S_{10}	0	1	1	0	0	1
		1	0	0	1	1	0
	S_5-S_{11}	0	1	1	0	ϕ	1
		1	0	0	1	ϕ	0
	S_5-S_{12}	0	1	1	0	0	0
		1	0	0	1	1	1
	S_5-S_{13}	0	1	1	0	ϕ	0
		1	0	0	1	ϕ	1
	S_5-S_{14}	0	1	1	0	0	ϕ
		1	0	0	1	1	ϕ
	S_5-S_{15}	0	1	1	0	ϕ	ϕ
		1	0	0	1	ϕ	ϕ

into a valid state to be tested. Since only one valid state of the switching element can be tested in a test phase, two test phases are needed for the switching element shown in Fig. 3.1. In phase 1, we test the valid state shown in Fig. 3.1(a) for all switching elements in the network and in phase 2, we test the valid state shown in Fig. 3.1(b). For a simple demonstration, consider a network with four terminal links in each side, as shown in Fig. 4.1. The labels on the input lines of the leftmost stage correspond to the required test vectors while the other labels indicate the fault-free response of the network to these test vectors. The fault-free response of the network shown in Fig. 4.1 assures that each one of the switching elements has its two input lines labelled with two different test vectors. These two different test vectors are exactly the test vectors needed to test each valid state of a switching element (see Section 4.1). These test vectors appearing on the input lines constitute a test set which can efficiently test all switching elements in the network. An algorithm for generating such an efficient test set for a network of size N is described as follows:

Step 1: Label the top terminal link in the left side of the network with test vector t = 01.

Step 2: Assume the labelled terminal links are named 0,1,..., and m-1 from top to bottom and the next m unlabelled terminal links are named, from top to bottom, m,m+1,..., and 2m-1, where $1 \leq m < N$ and m is in 2's power. Label terminal link m+i with $\overline{L(i)}$ for $0 \leq i \leq m-1$, where L(i) is the test vector assigned to terminal link i and $\overline{L(i)}$ is the complement.

Step 3: For the unlabelled terminal links in the left side, repeat Step 2 until all N terminals are labelled.

The test set generated by the above algorithm is good for both test phases and an example is given in Fig. 4.2, which shows the fault free response of an example network to the test vectors generated by the above algorithm. It can be seen that to detect single faults in a network four tests (two for each test phase)

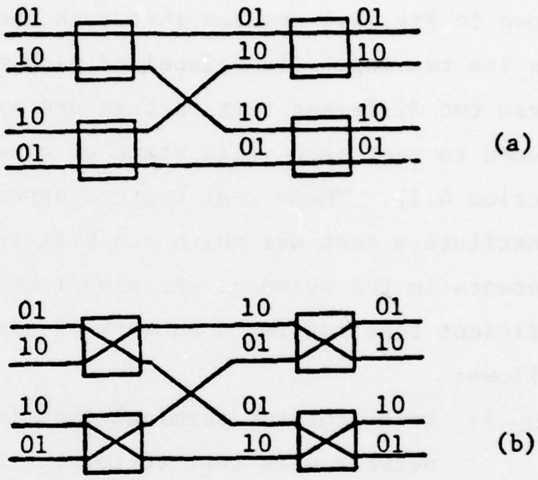


Fig. 4.1 Test set and response for the basic composite network. (a) Phase 1 test. (b) Phase 2 test.

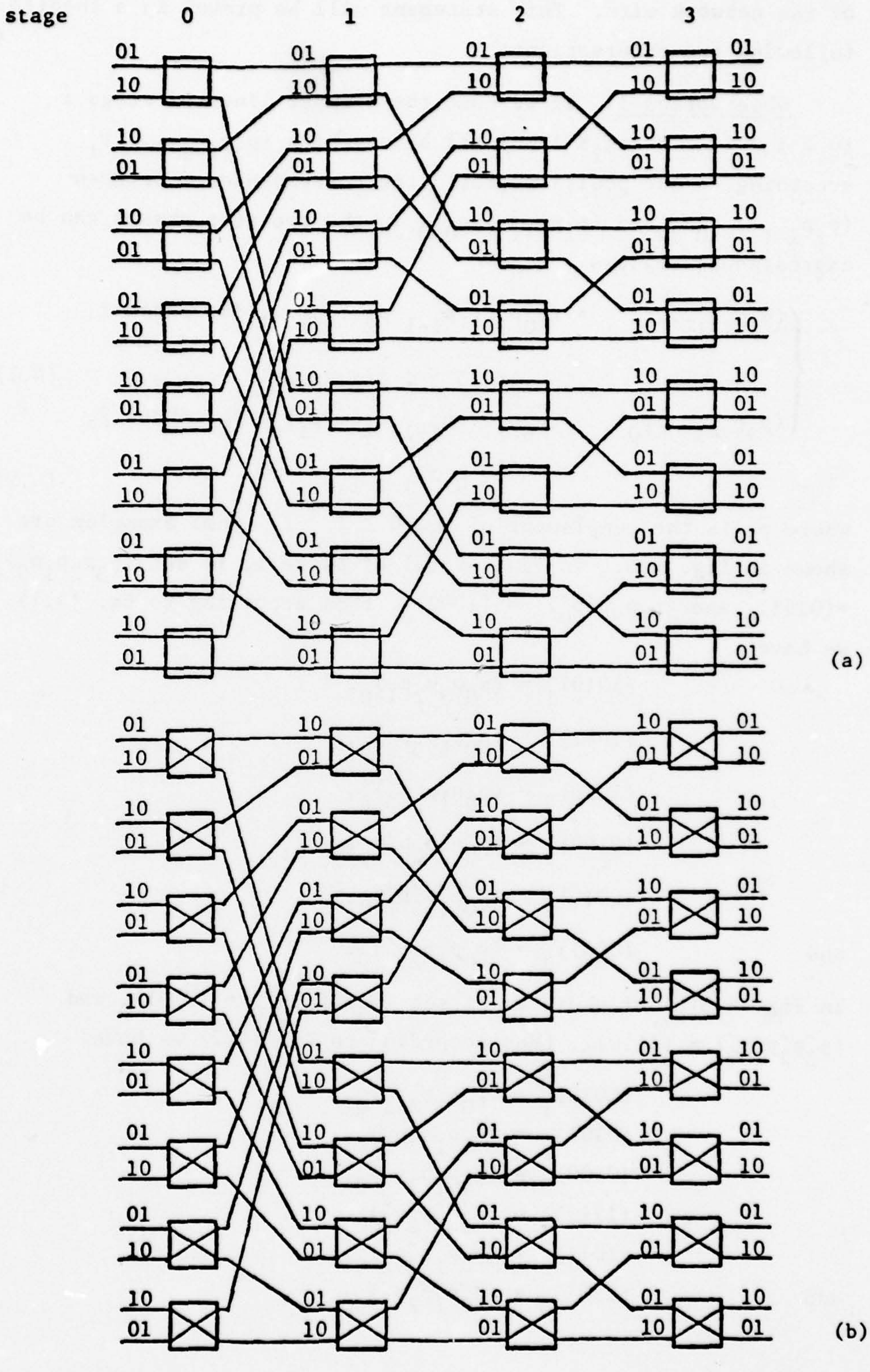


Fig. 4.2 Fault-free response of a network to the test set.
(a) Phase 1 test. (b) Phase 2 test.

are necessary and sufficient and the test length is independent of the network size. This statement will be proven in a theorem following two observations:

Observation 1: If we name the N input lines of stage i ($0 \le i \le \ell$, $\ell = \log_2 N-1$) by $\ell+1$ binary bits $(p_\ell p_{\ell-1} \cdots p_0)_i$ according to the position order, the correspondence between $(p_\ell p_{\ell-1} \cdots p_0)_i$ and $(p_\ell p_{\ell-1} \cdots p_0)_0$ in the two test phases can be expressed as follows:

$$\left\{\begin{array}{ll} (p_\ell p_{\ell-1} \cdots p_0)_i = (p_0 p_1 \cdots p_{i-1}\ p_\ell \cdots p_i)_0, & \text{for phase 1,} \\ & 0 \le i \le \ell, \text{ and} \qquad (4.1) \\ (p_\ell p_{\ell-1} \cdots p_0)_i = (\overline{p}_0 \overline{p}_1 \cdots \overline{p}_{i-1}\ p_\ell \cdots p_i)_0, & \text{for phase 2,} \end{array}\right.$$

$$0 \le i \le \ell, \qquad (4.2)$$

where $\overline{p}_j$ is the complement of p_j, $0 \le j < i$. Some examples are shown in Fig. 4.3. In Fig. 4.3(a) of phase 1, if set $(p_3 p_2 p_1 p_0)_0 =(0101)_0$ and $(p_3' p_2' p_1' p_0')_0 = (1000)_0$, then according to Eq. (4.1) we have

$$(1010)_1 = (p_0 p_3 p_2 p_1)_0,$$

$$(1001)_2 = (p_0 p_1 p_3 p_2)_0,$$

$$(1010)_3 = (p_0 p_1 p_2 p_3)_0,$$

$$(0100)_1 = (p_0' p_3' p_2' p_1')_0,$$

$$(0010)_2 = (p_0' p_1' p_3' p_2')_0,$$

and $$(0001)_3 = (p_0' p_1' p_2' p_3')_0.$$

In Fig. 4.3(b) of phase 2, if set $(p_3 p_2 p_1 p_0)_0 = (0101)_0$ and $(p_3' p_2' p_1' p_0') = (1110)_0$, then according to Eq. (4.2) we have

$$(0010)_1 = (\overline{p}_0 p_3 p_2 p_1)_0,$$

$$(0101)_2 = (\overline{p}_0 \overline{p}_1 p_3 p_2)_0,$$

$$(0100)_3 = (\overline{p}_0 \overline{p}_1 \overline{p}_2 p_3)_0,$$

$$(1111)_1 = (\overline{p}_0' p_3' p_2' p_1')_0,$$

$$(1011)_2 = (\overline{p}_0' \overline{p}_1' p_3' p_2')_0,$$

and $$(1001)_3 = (\overline{p}_0' \overline{p}_1' \overline{p}_2' p_3')_0.$$

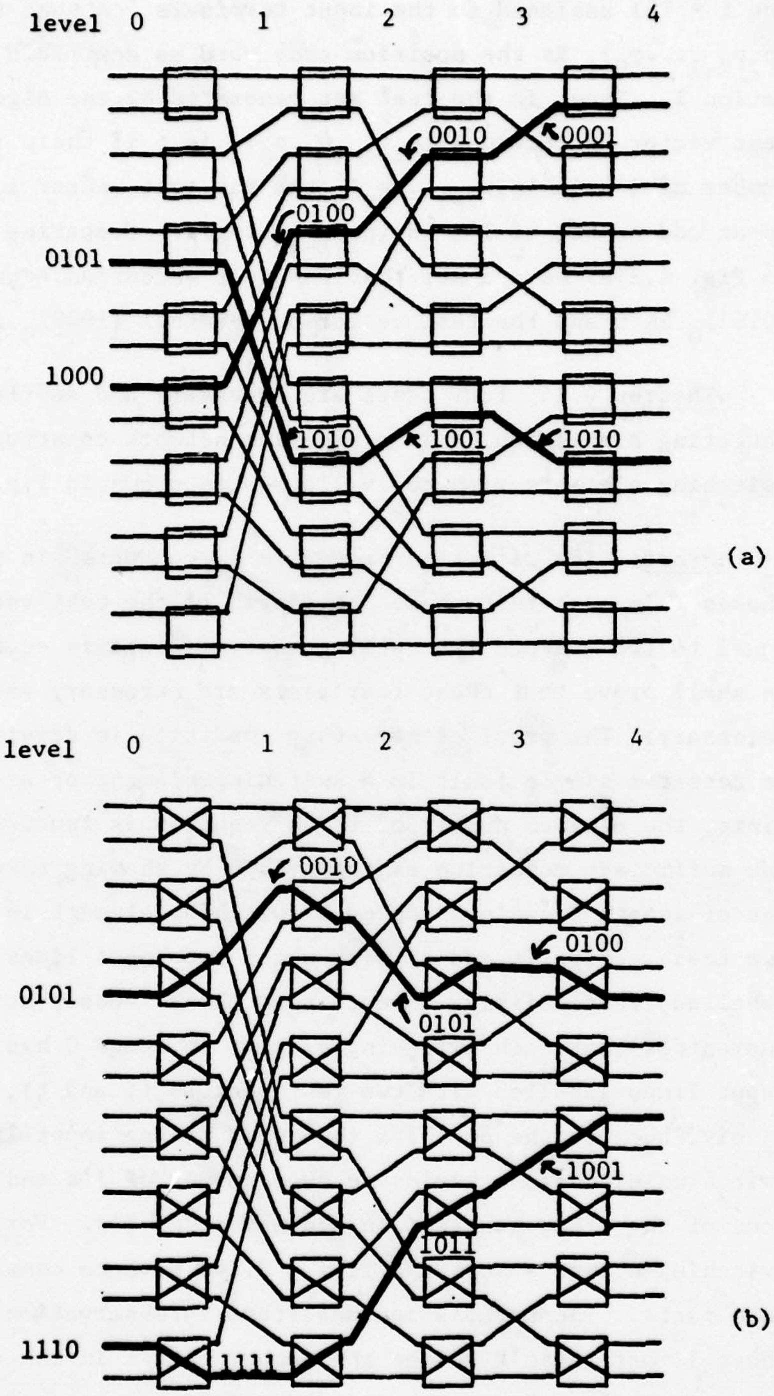


Fig. 4.3 Examples of observations. (a) Phase 1. (b) Phase 2.

<u>Observation 2</u>: There are two kinds of test vectors ($t = 01$ and $\overline{t} = 10$) assigned to the input terminals. Assume that $(p_{\ell}p_{\ell-1}\cdots p_0)_i$ is the position code word as described in Observation 1. Then, in the test set generated by the algorithm, the test vector on terminal $(p_{\ell}p_{\ell-1}\cdots p_0)_0$ is t if there is an even number of 1's in $(p_{\ell}p_{\ell-1}\cdots p_0)_0$ and the test vector is $\overline{t}$ if there is an odd number of 1's in $(p_{\ell}p_{\ell-1}\cdots p_0)_0$. Comparing Fig. 4.3(a) to Fig. 4.2(a) we can see that the test vector on terminal $(0101)_0$ is t and the test vector on terminal $(1000)_0$ is $\overline{t}$.

<u>Theorem 4.1</u>: Four tests are necessary and sufficient for detecting single faults in a baseline network constructed of switching elements with two valid states shown in Fig. 3.1.

<u>Proof</u>: The detection procedure is conducted in two test phases. In each test phase the length of the test vectors is equal to two. Hence the total number of tests is equal to four. We shall prove that these four tests are necessary and sufficient. Necessary: The proof of necessary condition is trivial since to detect a single fault in a switching element or at related links, the minimum number of tests required is four. Sufficient: The sufficient condition can be proved by showing that the test set of length 2 can provide each switching element in the network two test vectors (t and $\overline{t}$) with which two input lines are labelled, respectively, in each test phase. Observation 2 guarantees that each switching element in stage 0 has its two input lines labelled with two test vectors (t and $\overline{t}$), respectively, because the position code word of one input line of these switching elements contains an even number of 1's and the position code of the other contains an odd number of 1's. For the switching elements in stage i, $i > 0$, we have to consider some more facts. The permutation described in Observation 1 for phase 1 test doesn't change the number of 1's in the code point $(p_{\ell}p_{\ell-1}\cdots p_0)_0$. Hence the statement in Observation 2 is also true for the input lines of stage i, $i > 0$, in the network. This assures that each switching element in the network has its two input lines labelled with two test vectors (t and $\overline{t}$), respectively,

in phase 1 set-up. However, in phase 2 set-up we have a modification. The permutation described in Observation 1 for phase 2 test could change the number of 1's in the code word $(p_\ell p_{\ell-1}\cdots p_0)_0$. It can be seen that if the stage number i is even, both $(p_\ell p_{\ell-1}\cdots p_0)_0$ and $(\bar{p}_0\bar{p}_1\cdots\bar{p}_{i-1}\; p_\ell\cdots p_i)_0$ have an even or odd number of 1's and if the stage number i is odd, $(p_\ell p_{\ell-1}\cdots p_0)_0$ has an even number of 1's while $(\bar{p}_0\bar{p}_1\cdots\bar{p}_{i-1}\; p_\ell\cdots p_i)_0$ has an odd number of 1's and vice versa. Hence we observed that the test vector for $(p_\ell p_{\ell-1}\cdots p_0)_i$ is t if the sum of i and the number of 1's in $(p_\ell p_{\ell-1}\cdots p_0)_i$ is even and the test vector is $\bar{t}$ if the sum is odd. The above observation also assures that each switching element in stage i, $i > 0$, in the network has its two input lines labelled with two test vectors (t and $\bar{t}$), respectively, in phase 2 set-up. Q.E.D.

B. Fault Location

The problem of locating a single fault can be partitioned into two subproblems: one is to locate the stuck fault at a link and the other is to locate the fault in a switching element.

1. Link stuck fault:

The first subproblem can easily be solved due to the following two observations:

Observation 3: A stuck fault at a link can cause one and only one faulty output at an observable terminal in each test phase. Each faulty output should be equal to either 00 or 11.

Observation 4: Each link in the network can uniquely be identified by two paths, one from phase 1 and another from phase 2.

The method to compute the link set in a path can be found in Chapter 3. The test set derived for detecting single faults can be used to locate a single stuck fault at a link. However, the link stuck fault may not be distinguishable from some switching element faults which will be discussed later. In spite of this indistinguishability, there exists a one-to-one relationship between a link stuck fault and a faulty output pattern.

Theorem 4.2: There is a one-to-one correspondence between the link stuck fault and the faulty output pattern. The faulty output pattern is a necessary condition of the link stuck fault.

Proof: Since there are $1+\log_2 N$ levels of links and each level contains N links, the total number of link-stuck-fault locations is equal to $N(1+\log_2 N)$. From Observations 3 and 4, a link stuck fault can be identified by using a faulty output at the phase 1 test and another faulty output at the phase 2 test, but not every pair of faulty outputs can necessarily pinpoint a link stuck fault. There are exactly $N(1+\log_2 N)$ pair of faulty outputs, each of which is correspondent to a link stuck fault. The correspondence is described as follows. The stuck fault of link $(p_\ell p_{\ell-1} \cdots p_0)_i$ can be pinpointed by the phase 1 faulty output at terminal link $(p_\ell \cdots p_{\ell-i+2} p_0 p_1 \cdots p_{\ell-i+1})_{\ell+1}$ for $0 < i \leq \ell+1$ or $(p_0 p_1 \cdots p_\ell)_{\ell+1}$ for $i = 0$ and the phase 2 faulty output at terminal link $(p_\ell \cdots p_{\ell-i+2} p_0 \bar{p}_1 \cdots \bar{p}_{\ell-i+1})_{\ell+1}$ for $0 < i \leq \ell+1$ or $(\bar{p}_0 \bar{p}_1 \cdots \bar{p}_\ell)_{\ell+1}$ for $i = 0$. The fault output at these two terminals should be equal (either 00 or 11). However this faulty output pattern is only a necessary condition for the link stuck fault since the same faulty output pattern may imply a switching element fault which will be discussed later. Q.E.D.

Example: A network along with its test responses of both phase 1 and phase 2 tests is shown in Fig. 4.4. Comparing the test outputs to the fault free output, we observe that the path of link set {6,6,3,5,6} in the phase 1 test and the path of link set {7,6,2,0,1} in the phase 2 test lead to the faulty output 00 of link $(0110)_4$ at the phase 1 test and $(0001)_4$ at the phase 2 test. Intersecting the two link sets, we can locate the fault at link $(0110)_1$, which is stuck at zero.

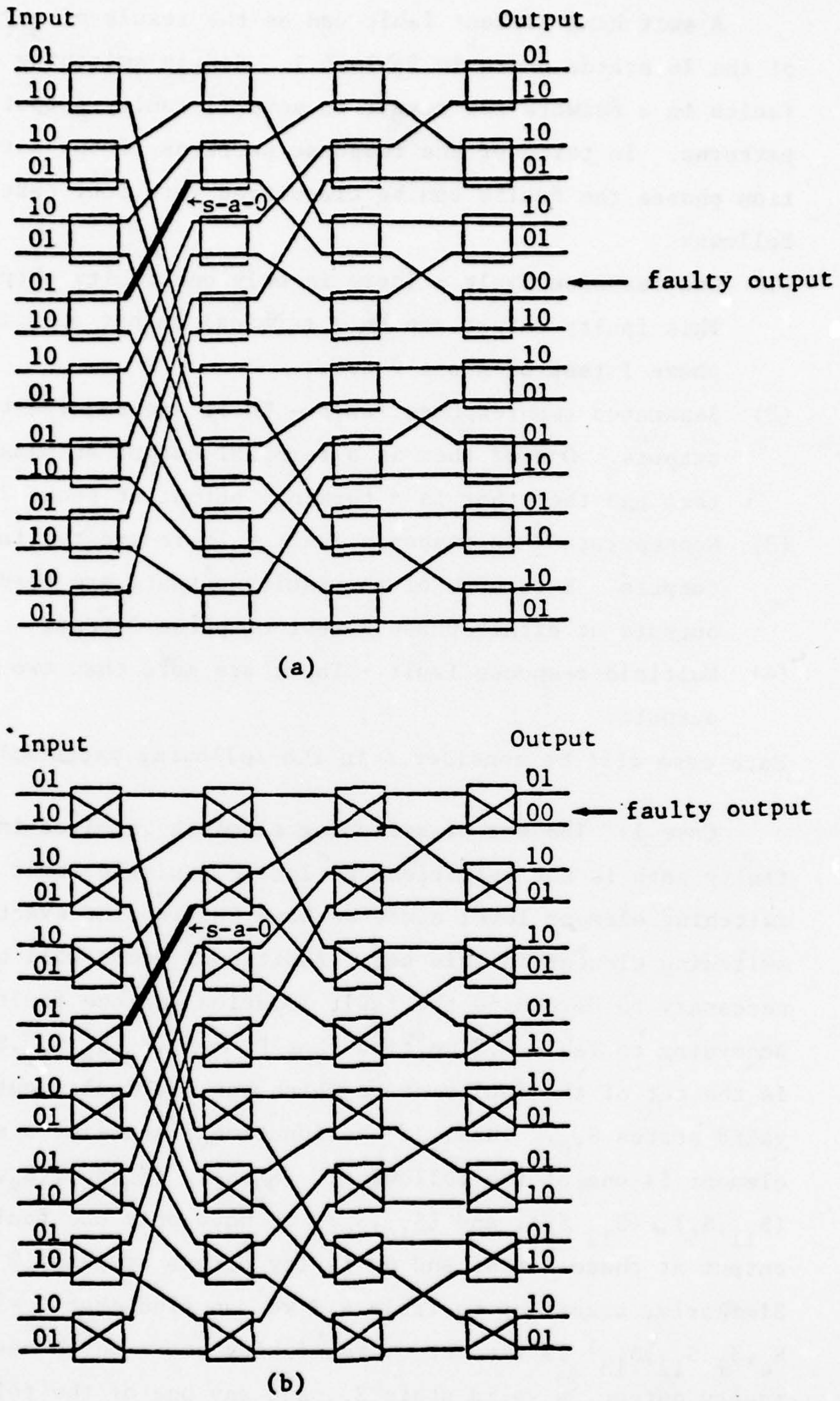


Fig. 4.4 Locating the link stuck fault. (a) Phase 1 test. (b) Phase 2 test.

2. Switching element fault:

A switching element fault can be the result of any one of the 16 states shown in Table 4.1. Single switching element faults in a network can result in several faulty output patterns. In terms of the response patterns of the detection phases the faults can be classified into four cases as follows:

(1) One-response fault - There is only one faulty output. This faulty output can be a terminal output at either phase 1 test or phase 2 test;

(2) Separated two-response fault - There are two faulty outputs. One of them is a terminal output at phase 1 test and the other is a terminal output at phase 2 test;

(3) Nonseparated two-response fault - There are two faulty outputs. But, both of the faulty outputs are terminal outputs at either phase 1 test or phase 2 test;

(4) Multiple-response fault - There are more than two faulty outputs.

Each case will be considered in the following paragraphs.

Case 1: The set of switching elements involved in a faulty path is not sufficient to locate a single fault at the switching element level since we have to pinpoint exactly one switching element in this set. Additional tests will be necessary to determine the fault location and the fault type. According to Table 4.2 we find that $P = \{S_2,S_3,S_8,S_{11},S_{12},S_{14}\}$ is the set of the faulty state which has one faulty output in valid states S_{10}. Thus, if the functional state of a switching element is one of the following: (S_2,S_5), (S_3,S_5),(S_8,S_5), (S_{11},S_5), (S_{12},S_5), and (S_{14},S_5), we have only one faulty output at phase 1 test and no faulty output at phase 2 test. Similarly, according to Table 4.3 we can find that $Q =\{S_1,S_3,$ $S_4,S_7,S_{12},S_{13}\}$ is the set of the faulty state which has one faulty output in valid state S_5, and any one of the following functional states: (S_{10},S_1), (S_{10},S_4), (S_{10},S_7), (S_{10},S_{12}), and (S_{10},S_{13}) results in one faulty output at the phase 2 test and no faulty output at the phase 1 test. These 12 fault types and the related faulty output patterns are shown in Table 4.4 which will be used to facilitate the proof of Theorem 4.3.

Theorem 4.3: The fault location and the fault type of the one-response fault can be determined by at most $6+2\lceil\log_2(\log_2 N)\rceil$ or $6+2\lfloor\log_2(\log_2 N)\rfloor$ tests.

Proof: The proof will be done by constructing an algorithm to determine the fault location and the fault type. The algorithm makes use of the principle of a binary search.

Assume that each stage of the network is a leaf of a binary tree in which each node has two incident branches, L and R. Starting at the root, the algorithm will generally try to pinpoint a fault in one of the branches until the pinpointed faulty branch is a leaf. In order to decide in which branch, L or R, the fault stays, an experiment is conducted at each node in the faulty tree path which is the path from the root to the faulty leaf. The experiment is a test in which the valid state of switching elements in L and the left of L should be set up as the one set up in the test which shows a faulty output and the valid state of switching elements in R and the right of R should be set up as the one set up in the test which shows the normal response. The two input vectors in the experiment are the same as those two at the detection phase. If the response of the experiment is not equal to the normal response, R branch is fault-free. Otherwise L branch is fault-free. The number of the nodes in the path, from root to leaf, of the tree is equal to either $\lceil\log_2(\log_2 N)\rceil$ or $\lfloor\log_2(\log_2 N)\rfloor$ where N is the number of terminal links in one side of the network, $\lceil\log_2(\log_2 N\rceil$ is the smallest integer which is greater than or equal to $\log_2(\log_2 N)$ and $\lfloor\log_2(\log_2 N)\rfloor$ is the largest integer which is less than or equal to $\log_2(\log_2 N)$. Hence we have to do either $\lceil\log_2(\log_2 N)\rceil$ or $\lfloor\log_2(\log_2 N)\rfloor$ experiments. In addition, four tests are used at the detection phase as described in Section 4.2.A. Thus, the total number of tests

Table 4.4 Faulty Output Pattern in Case 1.

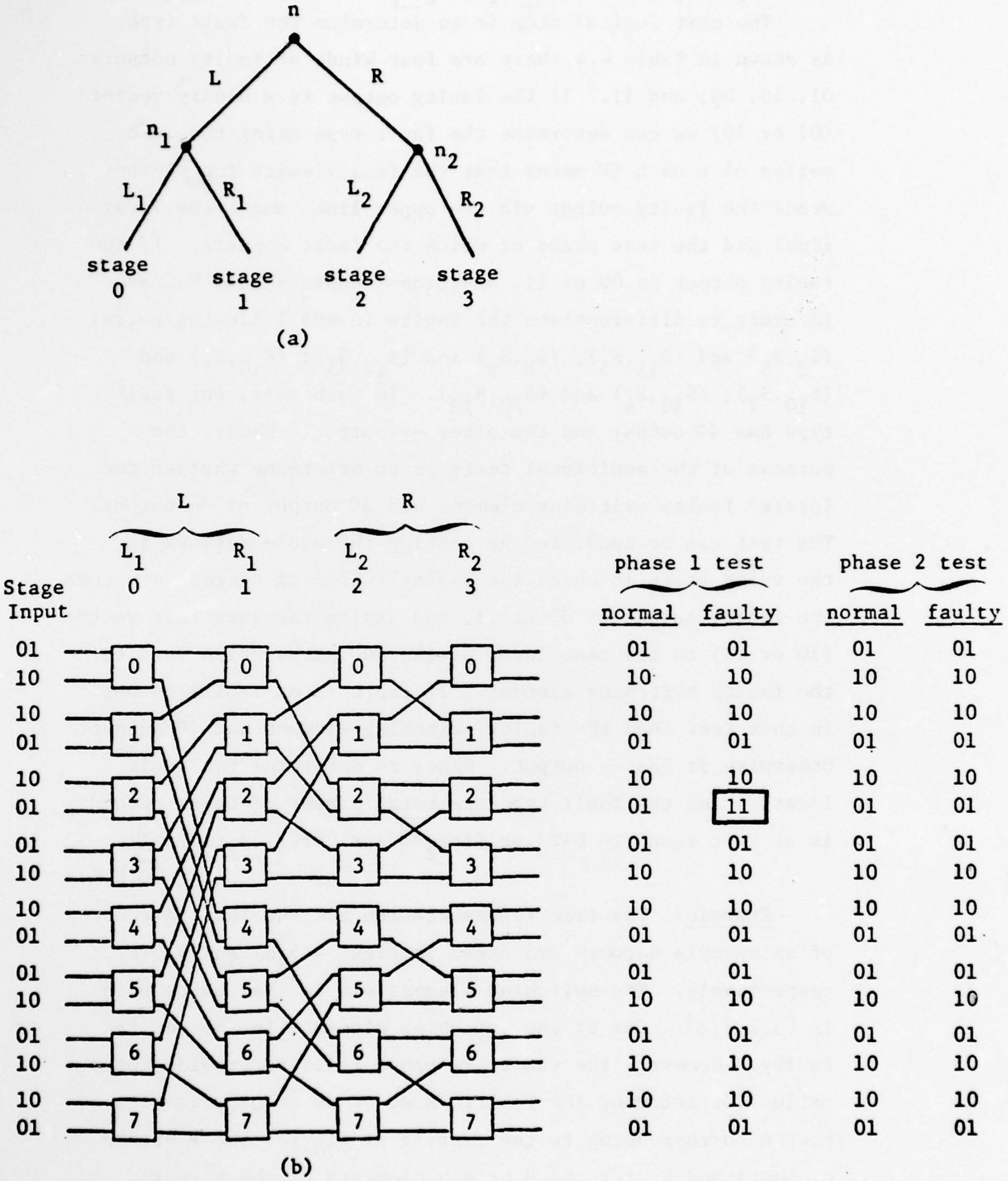
Faults of Case 1	Upper (U) or Lower (L) Link by Which the Faulty Switching Element Sends the Fault	Test Phase at Which Fault Appears.	Faulty Output --: (00 or 11) ϕϕ: (00 or 11) Binary Vector (10 or 01)
(S_2,S_5)	U	1	--
(S_3,S_5)	U	1	Binary Vector
(S_8,S_5)	L	1	--
(S_{11},S_5)	U	1	ϕϕ
(S_{12},S_5)	L	1	Binary Vector
(S_{14},S_5)	L	1	ϕϕ
(S_{10},S_1)	L	2	--
(S_{10},S_3)	L	2	Binary Vector
(S_{10},S_4)	U	2	--
(S_{10},S_7)	L	2	ϕϕ
(S_{10},S_{12})	U	2	Binary Vector
(S_{10},S_{13})	U	2	ϕϕ

for locating a single fault in this case is equal to $4+2\lceil \log_2(\log_2 N)\rceil$ or $4+2\lfloor \log_2(\log_2 N)\rfloor$.

The next logical step is to determine the fault type. As shown in Table 4.4 there are four kinds of faulty outputs: 01, 10, 00, and 11. If the faulty output is a binary vector (01 or 10) we can determine the fault type using the information of U or L (U means that the faulty switching element sends the faulty output via its upper link, and L the lower link) and the test phase at which the fault appears. If the faulty output is 00 or 11, additional tests should be made in order to differentiate the faults in the following pairs: (S_2,S_5) and (S_{11},S_5); (S_8,S_5) and (S_{14},S_5); (S_{10},S_1) and (S_{10},S_7); (S_{10},S_4) and (S_{10},S_{13}). In each pair, one fault type has $\phi\phi$ output and the other -- output. Hence, the purpose of the additional tests is to determine whether the located faulty switching element had $\phi\phi$ output or -- output. The test can be conducted by setting the whole network in the valid state in which the faulty switching element presents the faulty output of 00 or 11, and assign the same test vector (10 or 01) to the test input of the two paths which lead to the faulty switching element. If there is no faulty output in this test then the faulty switching element had $\phi\phi$ output. Otherwise it has -- output. Hence to determine the fault location and the fault type the total number of tests needed is at most equal to $6+2\lceil \log_2(\log_2 N)\rceil$ or $6+2\lfloor \log_2(\log_2 N)\rfloor$.

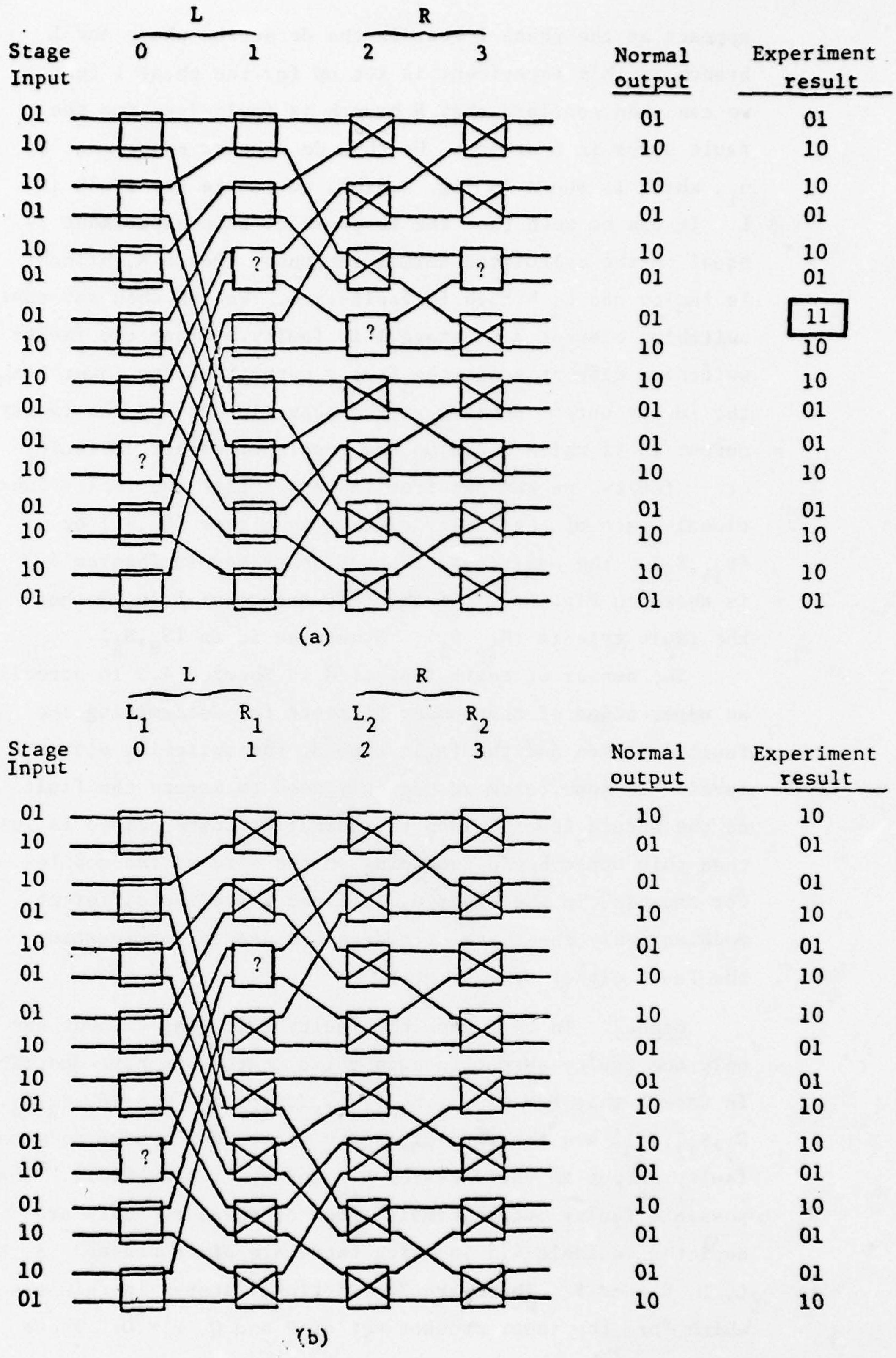
Q.E.D.

<u>Example</u>: The tree representation and the test response of an example network are shown in Figs. 4.5(a) and 4.5(b), respectively. The switching element set in the faulty path is {5,2,3,2}. One of the switching elements in the set is faulty. However, the faulty response fails to provide information for locating the fault. Now, we do an experiment at root n corresponding to the testing of S_{10} for the L branch elements and S_5 for the R branch elements of the network. The set-up and the experiment results which contain one faulty output are shown in Fig. 4.6(a). Since the faulty output



Input	phase 1 test normal	phase 1 test faulty	phase 2 test normal	phase 2 test faulty
01	01	01	01	01
10	10	10	10	10
10	10	10	10	10
01	01	01	01	01
10	10	10	10	10
01	01	$\boxed{11}$	01	01
01	01	01	01	01
10	10	10	10	10
10	10	10	10	10
01	01	01	01	01
01	01	01	01	01
10	10	10	10	10
01	01	01	01	01
10	10	10	10	10
10	10	10	10	10
01	01	01	01	01

Fig. 4.5 Case 1 (one faulty output). (a) Tree representation of the network. (b) Network structure and response of detection phases.



(a)

(b)

Fig. 4.6 Experiments in Case 1. (a) At node n. (b) At node n_1.

appears at the phase 1 test in the detection phase and L branch in this experiment is set up for the phase 1 test, we can then conclude that R branch is fault-free and the fault stays in L branch. We then do another experiment at n_1, which is shown in Fig. 4.6(b), to locate the fault in L. It can be seen that the response of this experiment is equal to the calculated normal response. Hence R_1 branch is faulty and L_1 branch is fault-free. We can then say that switching element 2 of stage 1 is faulty. Since the faulty switching element sends the faulty output via its lower link, the faulty output appears at the phase 1 test and the faulty output is 11 which could be the result of either $\phi\phi$ faulty or -- faulty, we can see from Table 4.4 that the faulty functional state of the faulty element is either (S_8,S_5) or (S_{14},S_5). The additional test as described in Theorem 4.3 is shown in Fig. 4.7. If the test output of Z is 10 then the fault type is (S_{14},S_5). Otherwise it is (S_8,S_5).

The number of tests indicated in Theorem 4.3 is actually an upper bound of the number of tests for determining the fault location and the fault type at the switching element level. In some cases we may only need to locate the fault at the module level. Then the number of tests needed is less than this upper bound depending on the size of the modules. For example, in the example, if L and R are two different modules, only the tests at node n are needed for locating the fault either at L or at R.

Case 2: In this case the faulty switching element has only one faulty output in each valid state. We have described in Case 1 that $P = \{S_2,S_3,S_8,S_{11},S_{12},S_{14}\}$ and $Q = \{S_1,S_3,S_4,S_7,S_{12},S_{13}\}$ are the sets of faulty states which have only one faulty output in valid states S_{10} and S_5, respectively. The possible faulty output combinations of these two sets are depicted in Table 4.5 in which there are six subcases: A, B, C, D, E, and F. There are 36 functional states in this case, which form the inner product set of P and Q, P × Q. These 36 functional states are classified into six subcases according

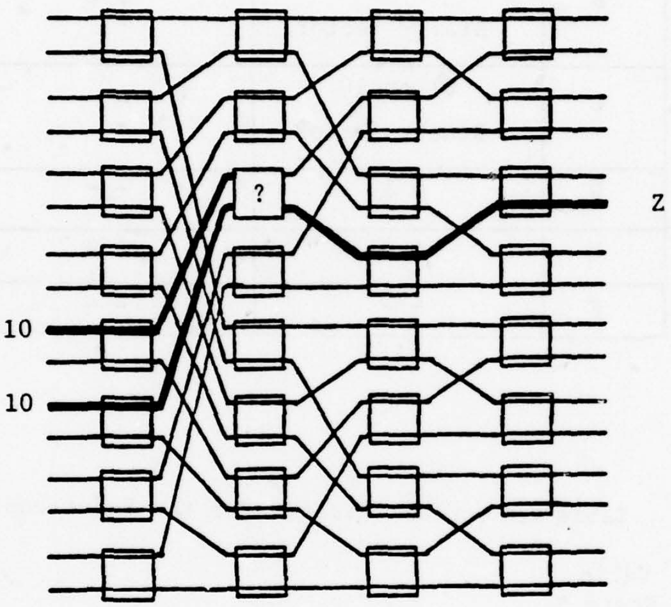


Fig. 4.7 Test to differentiate φφ and -- for the example in Case 1.

Table 4.5 Faulty Output Pattern in Case 2.

	Faulty Outputs	
Subcase	1	2
A	01 or 10 Binary Vector	01 or 10 Binary Vector
B	01 or 10 Binary Vector	ϕ ϕ
C	01 or 10 Binary Vector	- -
D	ϕ ϕ	- -
E	ϕ ϕ	ϕ ϕ
F	- -	- -

Table 4.6 Classification of the Functional State in Case 2.

Valid State S_{10} / Valid State S_5	S_{12}	S_3	S_2	S_8	S_{11}	S_{14}
S_{12}	A	A	C	C	B	B
S_3	A	A	C	C	B	B
S_4	C	C	F	F	D	D
S_1	C	C	F	F	D	D
S_{13}	B	B	D	D	E	E
S_7	B	B	D	D	E	E

to the faulty output patterns as shown in Table 4.5. The classification is shown in Table 4.6 in which the horizontal caption is for the faulty states of valid state S_{10} and the vertical for the faulty states of valid state S_5. An example of reading Table 4.6 is described below. Suppose (S_3,S_{13}) is a faulty functional state of (S_{10},S_5). The B at the intersection of column S_3 and row S_{13} in Table 4.6 implies that the switching element in functional state (S_3,S_{11}) will result in a faulty output of a binary vector in a test phase and a $\phi\phi$ faulty output in another test phase according to Table 4.5. Examples for the subcases are shown in Table 4.7. The examples show that there are two common switching elements in the two faulty paths. One of these two common switching elements is faulty. Additional test sets should be derived in order to locate the fault within these two questionable switching elements. In some examples not shown in Table 4.7 there is only one common switching element in the two faulty paths. In these examples the common switching element is either in the rightmost stage or in the leftmost stage.

Theorem 4.4: The fault location and the fault type of the Subcase A fault can be determined by at most 8 tests, independent of the network size.

Proof: The proof can be divided into two steps. The first step develops an algorithm to locate the fault at the switching element level and counts the number of tests needed in the algorithm. The second step counts the number of tests needed for determining the fault type in the subcase. We proceed to the first step after the Subcase A is identified by using the faulty output at the phase 1 test and the phase 2 test. If there is only one common switching element in the switching element sets of the faulty paths, the common switching element is faulty. If there are two questionable switching elements, a procedure should be performed to locate the fault. The procedure, named Algorithm A, is shown as follows:

Step 1: Subdivide the network into two subnetworks, L and R,

Table 4.7 Examples for Case 2 .

Subcase	A		B		C		D		E		F	
Faulty Functional State	(S_{12},S_3)		(S_3,S_{13})		(S_2,S_{12})		(S_{11},S_1)		(S_{11},S_{13})		(S_8,S_1)	
Phase	1	2	1	2	1	2	1	2	1	2	1	2
	01	01	01	01	01	01	01	01	01	01	01	01
T	10	10	$\boxed{01}$	10	$\boxed{--}$	10	10	10	$\boxed{\phi\phi}$	10	10	10
E	10	10	10	$\boxed{\phi\phi}$	10	$\boxed{01}$	10	10	10	$\boxed{\phi\phi}$	10	10
S	01	01	01	01	01	01	01	01	01	01	01	01
T	10	10	10	10	10	10	10	10	10	10	10	10
	$\boxed{10}$	01	01	01	01	01	$\boxed{\phi\phi}$	01	01	01	$\boxed{--}$	01
R	01	$\boxed{10}$	01	01	01	01	01	$\boxed{--}$	01	01	01	$\boxed{--}$
E	10	10	10	10	10	10	10	10	10	10	10	10
S	10	10	10	10	10	10	10	10	10	10	10	10
U	01	01	01	01	01	01	01	01	01	01	01	01
L	01	01	01	01	01	01	01	01	01	01	01	01
T	10	10	10	10	10	10	10	10	10	10	10	10
	01	01	01	01	01	01	01	01	01	01	01	01
	10	10	10	10	10	10	10	10	10	10	10	10
	10	10	10	10	10	10	10	10	10	10	10	10
	01	01	01	01	01	01	01	01	01	01	01	01
Element Sets of Faulty Paths	{5,2,3,2} {4,2,3,3}		{4,2,1,0} {5,2,1,1}		{4,2,1,0} {5,2,1,1}		{5,2,3,2} {4,2,3,3}		{4,2,1,0} {5,2,1,1}		{5,2,3,2} {4,2,3,3}	

with one of the two questionable switching elements found in the faulty paths in L and the other in R.

Step 2: Set up the switching elements in L in the valid state of an arbitrarily chosen test phase.

Step 3: Compute the link subsets of the paths leading to the input lines of the questionable switching element in R subnetwork. One of the two paths is a normal subpath.

Step 4: Use the same test vectors as those in detection phases except that of the normal path found in Step 3. Use the complement of the faulty output in the chosen test phase found in Step 2 as the test vector of the normal path.

Step 5: Test R in two valid states. If the two output vectors from the questionable switching element in R are equal to the normal ones, then R is fault-free. Otherwise L is fault-free.

The number of additional tests for locating such a single fault is equal to four.

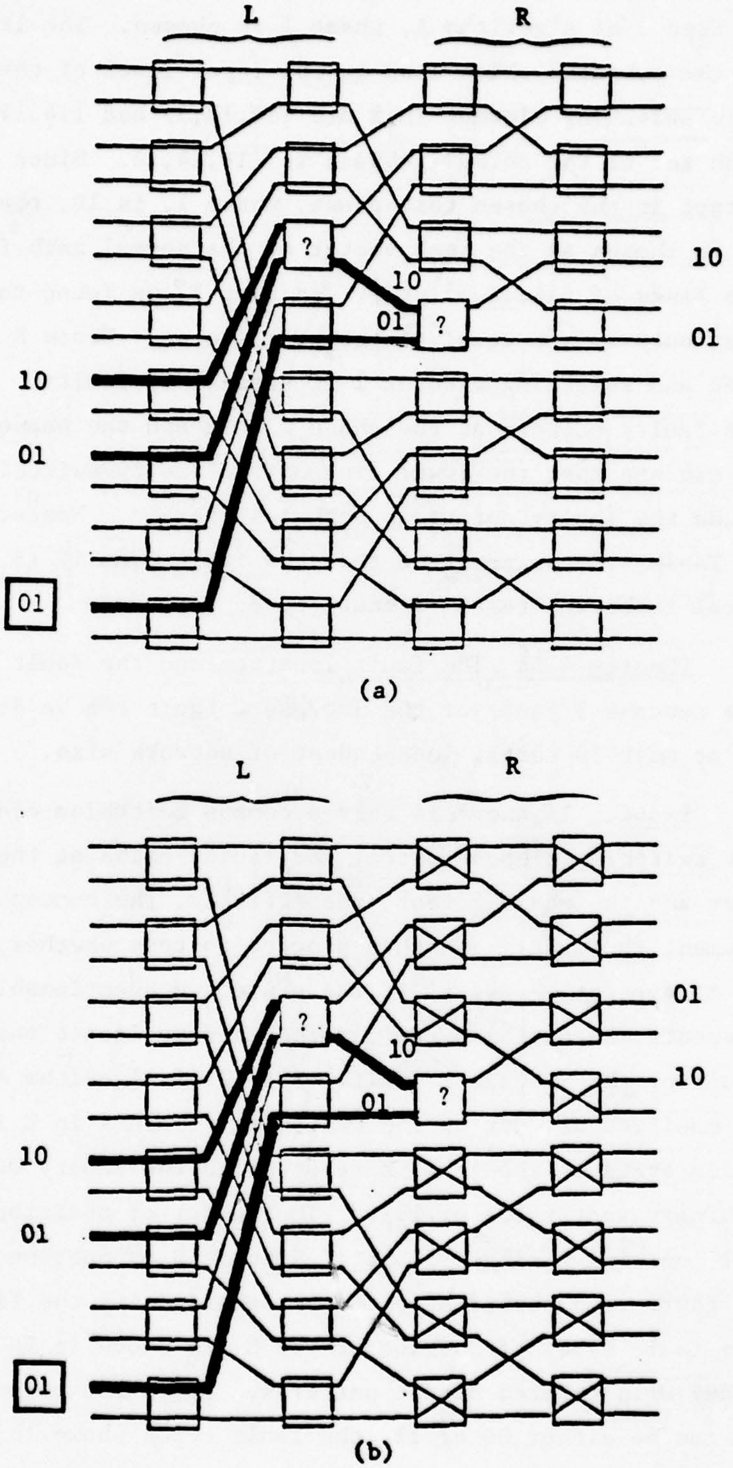
The second step is then to identify the fault type. As shown in Table 4.8, the four fault types of Subcase A can be differentiated by using the information of the link via which the faulty switching element sends the fault output. Hence the fault type can be determined, according to Table 4.8, by inspecting which link of the faulty switching element passes the faulty output to the terminal in each test phase. No additional test is needed.

The total number of tests required is equal to 4 or 8 which includes 4 for the detection phase and 4 for locating the fault. Q.E.D.

Example: The example for Subcase A in Table 4.7 is illustrated here. The network structure along with the test set-up is shown in Figs. 4.8(a) and (b). As shown in Table 4.7 there are two questionable switching elements: switching element 2 of stage 1 and switching elements 3 of stage 2.

Table 4.8 Faulty Output Pattern in Subcase A of Case 2.

Faults of Subcase A	Upper (U) or Lower (L) Link by Which the Faulty Switching Element Sends the Fault		Faulty Output Binary Vector (01 or 10)	
	Phase 1 Test	Phase 2 Test	Phase 1 Test	Phase 2 Test
(S_{12},S_{12})	L	U	Binary Vector	Binary Vector
(S_{12},S_3)	L	L	Binary Vector	Binary Vector
(S_3,S_{12})	U	U	Binary Vector	Binary Vector
(S_3,S_3)	U	L	Binary Vector	Binary Vector



(a)

(b)

Fig. 4.8 Subcase A. (a) Phase 1 test on R. (b) Phase 2 test on R.

The network is subdivided into L and R with switching element 2 of stage 1 in L and switching element 3 of stage 3 in R. In Step 2 of Algorithm A, phase 1 is chosen. The link subsets of the subpaths which lead to the input lines of the questionable switching element in R are {10,10,5} and {14,14,7}. The link set of the normal subpath is {14,14,7}. Since the faulty output in the chosen test phase, phase 1, is 10, the vector 01 is chosen as the test vector of the normal path in which the links of {14,14,7} stay. In Step 5, we found that the test outputs are equal to the normal ones. Hence R is fault-free and switching element 2 of stage 1 is faulty. Inspecting the faulty outputs at the phase 1 test and the phase 2 test we can see that the lower link of the faulty switching element sends the faulty output in both test phases. Hence, referring to Table 4.8, we conclude that the fault type if (S_{12},S_3). The total number of tests is equal to 8.

Theorem 4.5: The fault location and the fault type of the Subcase B fault or the Subcase C fault can be determined by at most 10 tests, independent of network size.

Proof: If there is only a common switching element in the switching element set of the faulty paths at the phase 1 test and the phase 2 test, respectively, the common switching element is faulty. We then proceed to test whether the fault is $\phi\phi$ type or -- type. If there are two questionable switching elements, Algorithm A can also be used to locate the Subcase B fault or the Subcase C fault. Step 2 of Algorithm A should be modified as "Set up the switching elements in L in the valid state of the test phase at which the faulty output is a binary vector (01 or 10)." The number of additional tests for locating a single fault of Subcase B or Subcase C is equal to four. The second step is then to identify the fault type. The fault types of Subcases B and C are shown in Table 4.9 along with related output patterns. Since the output of $\phi\phi$ or -- can be either 00 or 11, the fault types shown in Table 4.9 can be partitioned into eight sets of fault types: $\{(S_{12},S_{13})$,

Table 4.9 Faulty Output Pattern in Subcases B and C of Case 2.

Faults of Subcase B or C		Upper (U) or Lower (L) Link by Which the Faulty Switching Element Sends the Fault		Faulty Output: Binary Vector (01 or 10) ϕϕ (00 or 11) -- (00 or 11)	
		Phase 1 Test	Phase 2 Test	Phase 1 Test	Phase 2 Test
B	(S_{12},S_{13})	L	U	Binary Vector	ϕϕ
	(S_{12},S_{7})	L	L	Binary Vector	ϕϕ
	(S_{3},S_{13})	U	U	Binary Vector	ϕϕ
	(S_{3},S_{7})	U	L	Binary Vector	ϕϕ
	(S_{11},S_{12})	U	U	ϕϕ	Binary Vector
	(S_{11},S_{3})	U	L	ϕϕ	Binary Vector
	(S_{14},S_{12})	L	U	ϕϕ	Binary Vector
	(S_{14},S_{3})	L	L	ϕϕ	Binary Vector
C	(S_{12},S_{4})	L	U	Binary Vector	--
	(S_{12},S_{1})	L	L	Binary Vector	--
	(S_{3},S_{4})	U	U	Binary Vector	--
	(S_{3},S_{1})	U	L	Binary Vector	--
	(S_{2},S_{12})	U	U	--	Binary Vector
	(S_{2},S_{3})	U	L	--	Binary Vector
	(S_{8},S_{12})	L	U	--	Binary Vector
	(S_{8},S_{3})	L	L	--	Binary Vector

$(S_{12},S_4)\}$,$\{(S_{12},S_7),(S_{12},S_1)\}$,$\{(S_3,S_{13}),(S_3,S_4)\}$,$\{(S_3,S_7),$ $(S_3,S_1)\}$,$\{(S_{11},S_{12}),(S_2,S_{12})\}$,$\{(S_{11},S_3),(S_2,S_3)\}$,$\{(S_{14},S_{12}),$ $(S_8,S_{12})\}$, and $\{(S_{14},S_3),(S_8,S_3)\}$. The two fault types in each set, one with $\phi\phi$ output, and the other with -- output, are equivalent according to the output pattern and additional tests are needed to further differentiate them. The procedure as described in Theorem 4.3 needs two additional tests. Hence the total number of tests needed is equal to 6 or 10. Q.E.D.

Example: The example for Subcase B in Table 4.7 is illustrated here. The two questionable switching elements are switching element 2 of stage 1 and switching element 1 of stage 2. Figs. 4.9(a) and (b) show the network set-up required to locate the fault. In Step 2 of Algorithm A, the phase 1 test is chosen. The outputs of the test are normal. Hence switching element 2 of stage 1 in L is faulty. The faulty type is either (S_3,S_{13}) or (S_3,S_4) according to Table 4.9, since the faulty output is sent via the upper link in both test phases. The test to determine whether the fault type is (S_3,S_{13}) or (S_3,S_4) is shown in Fig. 4.10. Since the test output is a binary vector the faulty output at the phase 2 test is $\phi\phi$. Hence the fault type is (S_3,S_{13}). The total number of tests is equal to 10.

Theorem 4.6: The fault location and the fault type of the subcase D fault or the subcase E fault can be determined by at most 12 tests, independent of network size.

Proof: If there is only a common switching element in the switching element sets of the faulty path at the phase 1 test and the phase 2 test, respectively, the common switching element (it is in the leftmost or the rightmost stage) is faulty. We then proceed to test whether the fault is $\phi\phi$ type or -- type in each test phase. This procedure takes four tests. Hence the total number of tests needed is equal to 8 which includes four tests for the detection phase. If there are two common switching elements in the switching element set of the faulty path as shown in Table 4.7, we have to take

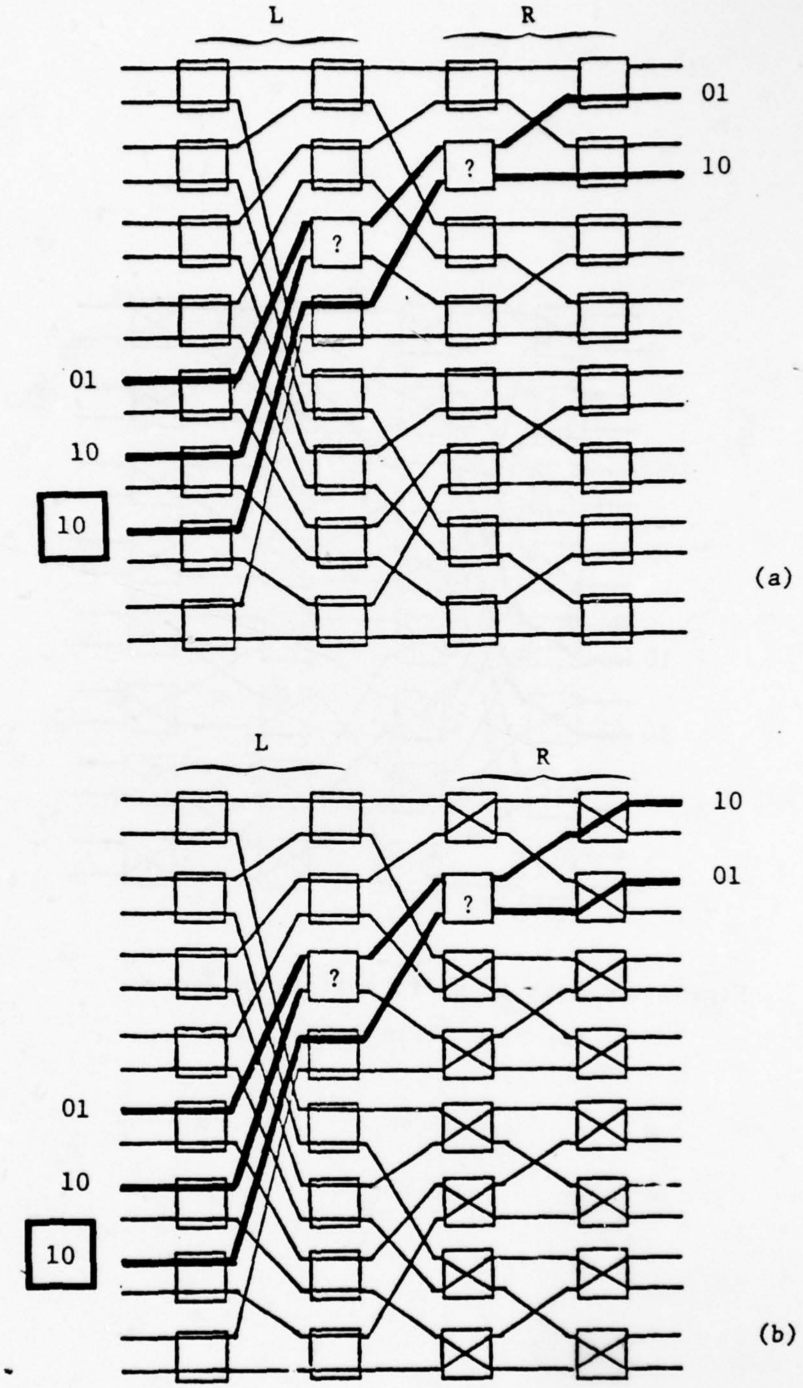


Fig. 4.9 Subcase B. (a) Phase 1 test. (b) Phase 2 test.

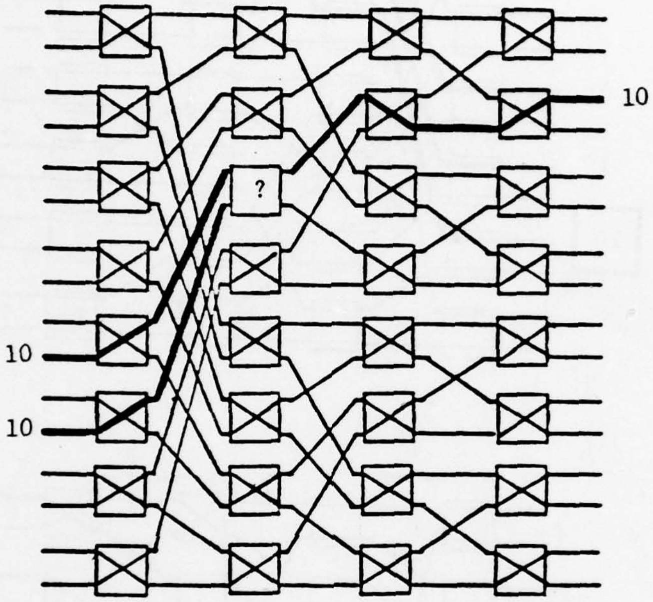


Fig. 4.10 **Test to differentiate φφ and --**
for example in Subcase B.

a procedure to locate the fault. The procedure, named Algorithm D, is shown as follows:

Step 1: Subdivide the network into L and R subnetworks with one of the two questionable switching elements in L and the other in R.

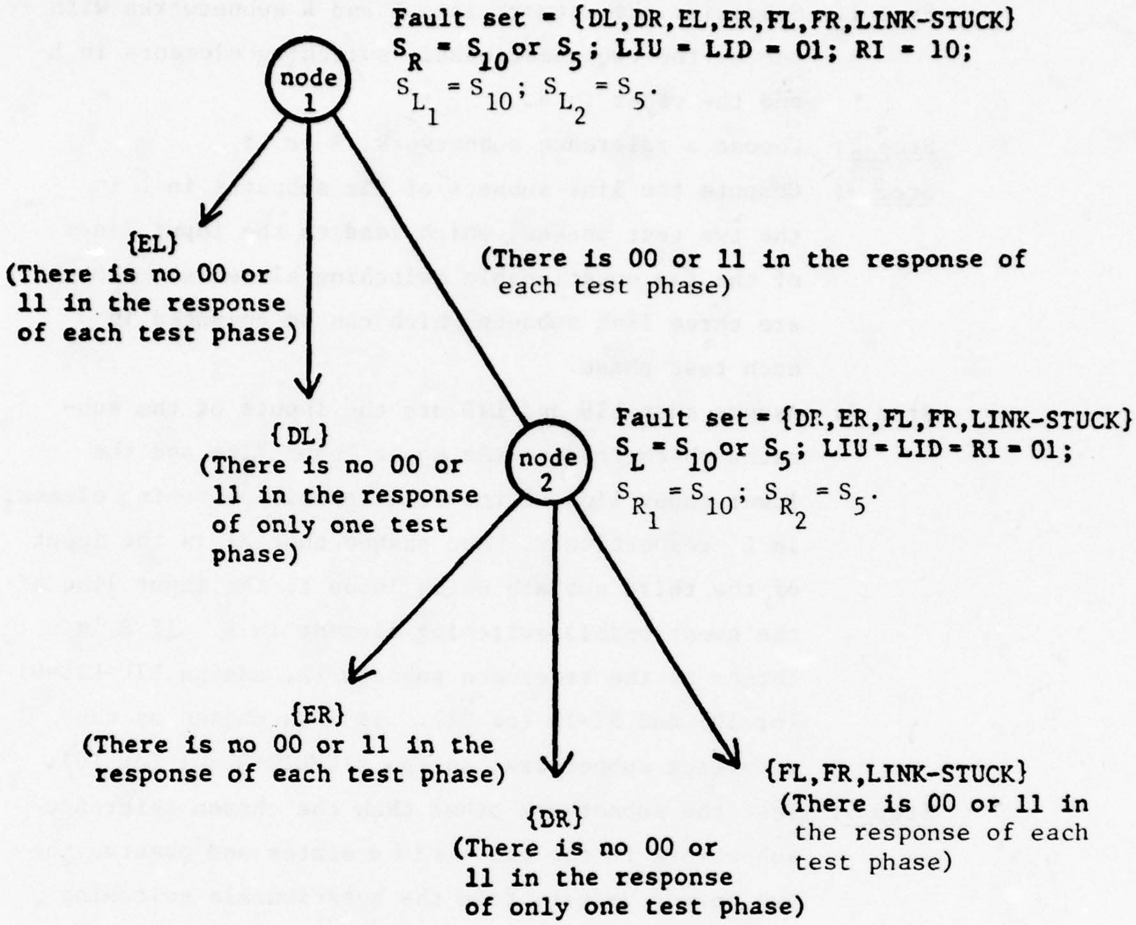
Step 2: Choose a reference subnetwork, R or L.

Step 3: Compute the link subsets of the subpaths in L in the two test phases, which lead to the input lines of the two questionable switching elements. There are three link subsets which can be computed in each test phase.

Step 4: Assume that LIU and LID are the inputs of the subpaths which lead to the upper input line and the lower input line of the questionable switching element in L, respectively. And assume that RI is the input of the third subpath which leads to the input line of the questionable switching element in R. If R is chosen as the reference subnetwork, assign LIU=LID=01 (or 10) and RI=10 (or 01). If L is chosen as the reference subnetwork, assign LIU=LID=RI=01 (or 10).

Step 5: Test the subnetwork other than the chosen reference subnetwork in the two valid states and observe the two output vectors from the questionable switching element in R in each test phase.

Depending on what we have observed in Step 5, we may have different decisions. The decision tree is shown in Fig. 4.11. Starting at node 1, we choose R as the reference subnetwork and process one cycle of Algorithm D. In Step 5, if there is no 00 or 11 in the test response of each test phase, the questionable switching element in L is faulty (Subcase E). Then according to the faulty output pattern shown in Table 4.10 for Subcase E we can determine the fault type. If there is no 00 or 11 in one test phase only, the questionable switching element in L is faulty (Subcase D). If only 00 or 11 appears at the phase 1 test of L, then the fault type is among (S_2,S_{13}), (S_2,S_7), (S_8,S_{13}), and (S_8,S_7),


Fault set = {DL,DR,EL,ER,FL,FR,LINK-STUCK}
$S_R = S_{10}$ or S_5; LIU = LID = 01; RI = 10;
$S_{L_1} = S_{10}$; $S_{L_2} = S_5$.
node 1
{EL}
(There is no 00 or 11 in the response of each test phase)
(There is 00 or 11 in the response of each test phase)
{DL}
(There is no 00 or 11 in the response of only one test phase)
Fault set = {DR,ER,FL,FR,LINK-STUCK}
$S_L = S_{10}$ or S_5; LIU = LID = RI = 01;
$S_{R_1} = S_{10}$; $S_{R_2} = S_5$.
node 2
{ER}
(There is no 00 or 11 in the response of each test phase)
{DR}
(There is no 00 or 11 in the response of only one test phase)
{FL,FR,LINK-STUCK}
(There is 00 or 11 in the response of each test phase)

<u>**Mnemonics not used in text:**</u>

DL: fault of subcase D in L	S_R: valid state of R
DR: fault of subcase D in R	S_L: valid state of L
EL: fault of subcase E in L	S_{R_1}: valid state of R in phase 1 test
ER: fault of subcase E in R	S_{R_2}: valid state of R in phase 2 test
FL: fault of subcase F in L	S_{L_1}: valid state of L in phase 1 test
FR: fault of subcase F in R	S_{L_2}: valid state of L in phase 2 test
LINK-STUCK: Link stuck fault	

Fig. 4.11 Summary of test procedures for Subcases D, E and F.

Table 4.10 Faulty Output Pattern in Subcases D and E of Case 2.

Faults of Subcases D or E		Upper (U) or Lower (L) Link by Which the Faulty Switching Element Sends the Fault		Faulty Output ϕϕ (00 or 11) -- (00 or 11)	
		Phase 1 Test	Phase 2 Test	Phase 1 Test	Phase 2 Test
D	(S_2,S_{13})	U	U	--	ϕϕ
	(S_2,S_7)	U	L	--	ϕϕ
	(S_8,S_{13})	L	U	--	ϕϕ
	(S_8,S_7)	L	L	--	ϕϕ
	(S_{11},S_4)	U	U	ϕϕ	--
	(S_{11},S_1)	U	L	ϕϕ	--
	(S_{14},S_4)	L	U	ϕϕ	--
	(S_{14},S_1)	L	L	ϕϕ	--
E	(S_{11},S_{13})	U	U	ϕϕ	ϕϕ
	(S_{11},S_7)	U	L	ϕϕ	ϕϕ
	(S_{14},S_{13})	L	U	ϕϕ	ϕϕ
	(S_{14},S_7)	L	L	ϕϕ	ϕϕ

and it can be determined by the faulty output patterns of the detection phase. If only 00 or 11 appears at the phase 2 test, then the fault type is among (S_{11},S_4), (S_{11},S_1), (S_4,S_4), and (S_4,S_1), and it can also be determined by the faulty output pattern of the detection phase. However, if there is 00 or 11 in the response of each test phase, we have to proceed to node 2. At node 2 we choose L as the reference subnetwork and go over Algorithm D once more. In this second run, we can make decisions according to the test response as we did at node 1, the decision could be one of the following: the fault of Subcase E is located in R if there is no 00 or 11 in the response of each test phase; the fault of Subcase D is located in R if there is no 00 or 11 in the response of only one test phase. The fault type can be determined in the same way as we did at node 1.

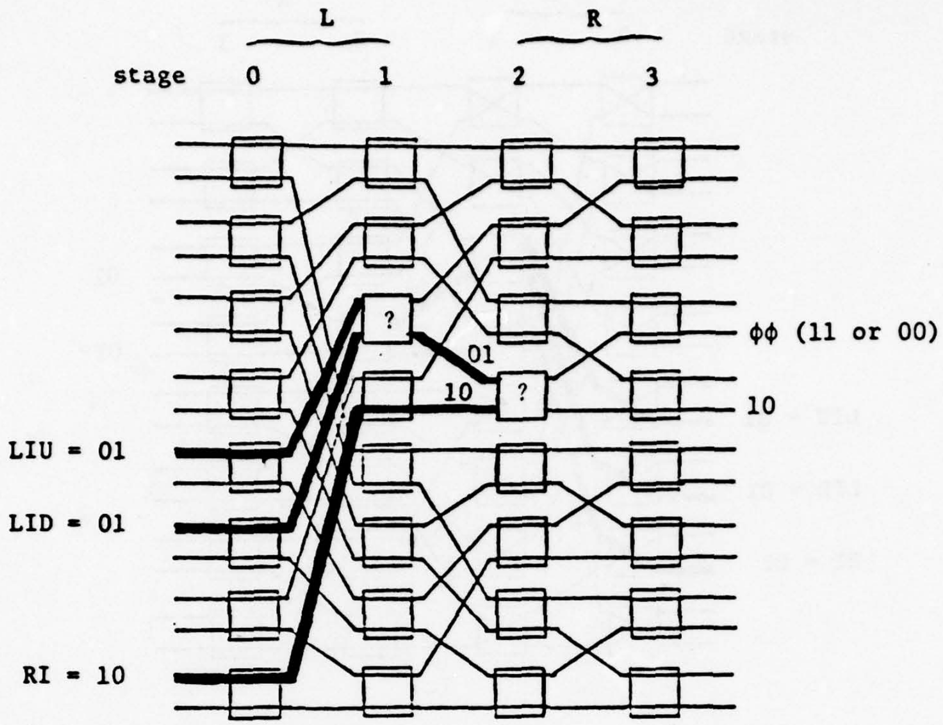
Hence the total number of tests needed to determine the fault location and the fault type is equal to 8 or 12.

Q.E.D.

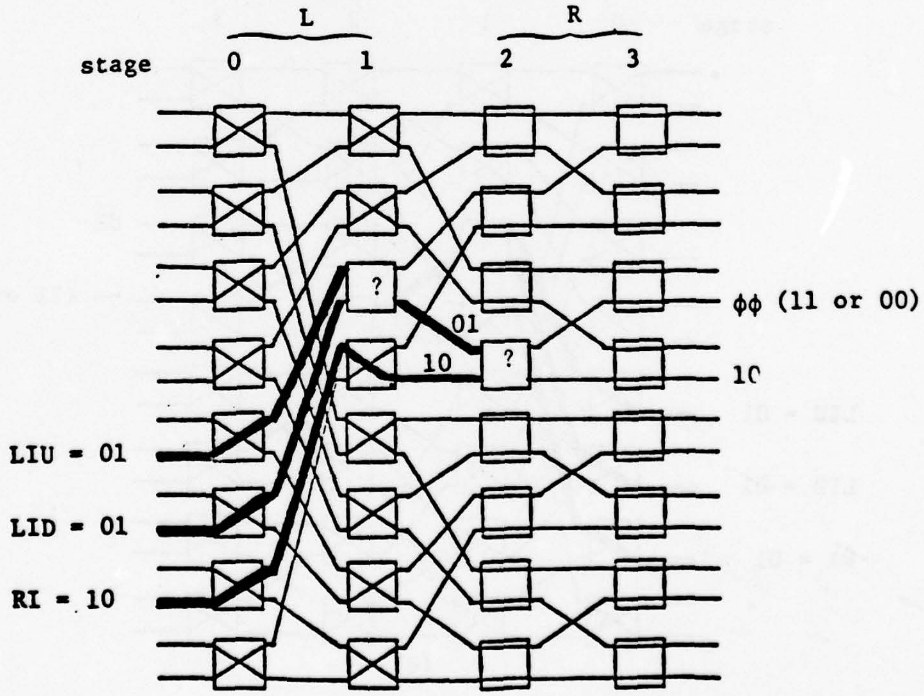
Example: The example for Subcase D in Table 4.7 is illustrated here. The structure of the example network and the associated test set-up are shown in Figs. 4.12(a), (b), (c), and (d). The two test phases at node 1 are shown in Figs. 4.12(a) and (b), respectively. Since there is 00 or 11 in the response of each test phase we proceed to node 2. The two test phases at node 2 are shown in Figs. 4.12(c) and (c), respectively. Since there is 00 or 11 at the phase 2 test and no 00 or 11 at the phase 1 test, we conclude that the fault is located at switching element 3 of stage 2 and the fault type is (S_{11},S_1) according to Table 4.10. The number of tests needed is equal to 12.

Remark: The fault of Subcase F cannot be pinpointed at the single switching element level and it is indistinguishable from a link stuck fault.

The faulty output pattern of Subcase F is shown in Table 4.11. If the fault can be located at the single switching element level, then no additional tests are needed for


L
R
stage
0
1
2
3
?
?
01
10
φφ (11 or 00)
10
LIU = 01
LID = 01
RI = 10

(a)


L
R
stage
0
1
2
3
?
?
01
10
φφ (11 or 00)
10
LIU = 01
LID = 01
RI = 10

(b)

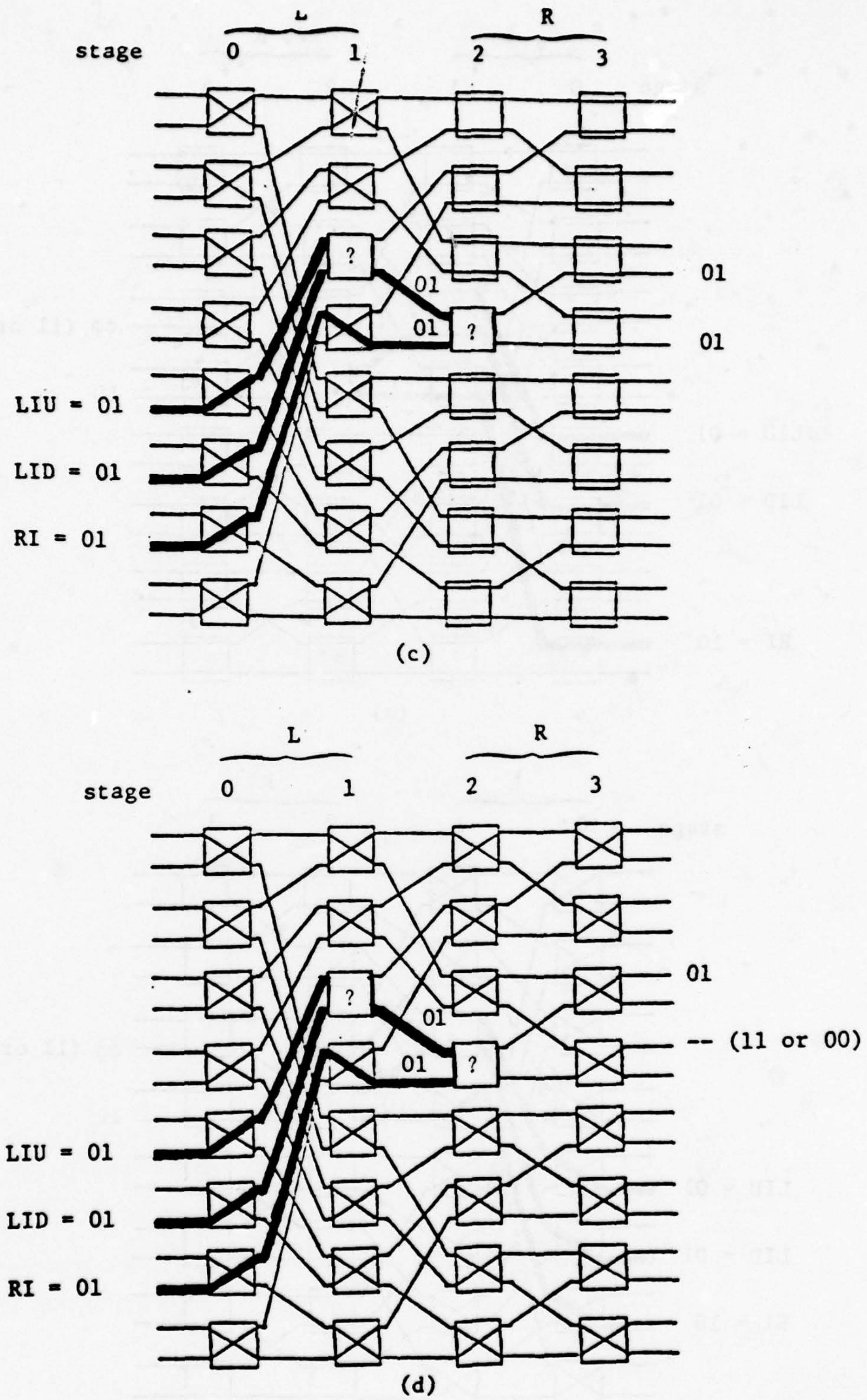


Fig. 4.12 **Subcase D.** **(a)** Node 1, $S_R=S_{10}$, $S_{L_1}=S_{10}$. **(b)** Node 1, $S_R=S_{10}$, $S_{L_2}=S_5$. **(c)** Node 2, $S_L=S_5$, $S_{R_1}=S_{10}$. **(d)** Node 2, $S_L=S_5$, $S_{R_2}=S_5$.

Table 4.11 Faulty Output Pattern in Subcase F of Case 2.

Faults of Subcase F	Upper (U) or Lower (L) Link by Which the Faulty Switching Element Sends the Fault		Faulty Output -- (00 or 11)	
	Phase 1 Test	Phase 2 Test	Phase 1 Test	Phase 2 Test
(S_2,S_4)	U	U	--	--
(S_2,S_1)	U	L	--	--
(S_8,S_4)	L	U	--	--
(S_8,S_1)	L	L	--	--

determining the fault type according to Table 4.11. However we can only locate the fault at eight location blocks shown in Fig. 4.13 using the faulty outputs of the detection phases. Figs. 4.13(a)-(d) show the location blocks of two common switching elements in the two faulty paths, and Figs. 4.13(e)-(h) show the location blocks of a common switching element in the two faulty paths, which should be in the rightmost stage (Figs. 4.13(e)-(f)) or the leftmost stage (Figs. 4.13(g)-(h)) of the network. The dash lines in Fig. 4.13 imply the possible faulty spots which include the link and the switching element. Because of the characteristics of -- fault it is impossible to further pinpoint the fault within each questionable location block by applying tests on the input side and observing output on the output side. Hence there exists an ambiguity between the link stuck fault and the Subcase F fault.

Case 3 and Case 4: We can compute the switching element sets of the faulty paths and the intersections of these sets should lead to a unique faulty switching element. In these two cases, no additional tests are required and only four tests which are developed for the detection phases are required for locating the fault.

Theorem 4.7: The fault location and the fault type of Case 3 or Case 4 can be determined by at most eight tests, independent of network size.

Proof: In Case 3, the faulty switching element has two faulty outputs at one of the test phases. There are 18 fault types in Case 2, which are the inner products of $\{S_0,S_1,S_4,S_5,S_6,S_7,S_9,S_{13},S_{15}\} \times \{S_5\}$ and $\{S_{10}\} \times \{S_0,S_2,S_6,S_8,S_9,S_{10},S_{11},S_{14},S_{15}\}$. Since the faulty switching elements can be uniquely identified by the switching element set of the two faulty paths in the same detection phase, the number of tests needed is equal to four. Two additional tests may be needed to differentiate $\phi\phi$ and -- as described in Theorem 4.3. In Case 4, there are 189 fault types and the fault type

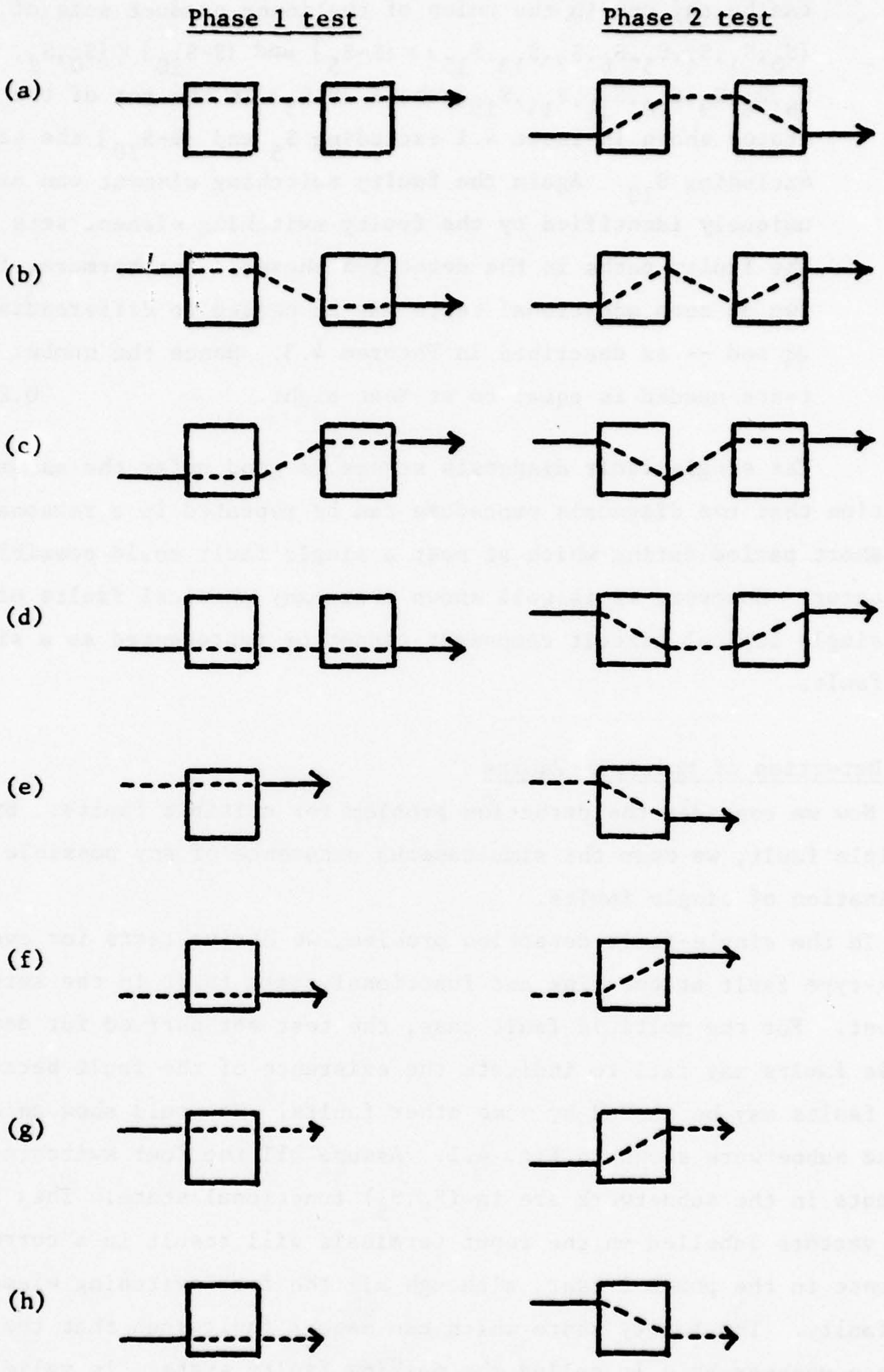


Fig. 4.13 Blocks of faulty location pattern of Subcase F (The dash line indicates the possible faulty spot).

can be any one in the union of the inner product sets of $\{S_0,S_1,S_4,S_5,S_6,S_7,S_{13},S_{15}\} \times \{S-S_5\}$ and $\{S-S_{10}\} \times \{S_0,S_2,S_6,S_8,S_9,S_{10},S_{11},S_{14},S_{15}\}$ where $\{S-S_5\}$ is the set of the states shown in Table 4.1 excluding S_5 and $\{S-S_{10}\}$ the set excluding S_{10}. Again the faulty switching element can be uniquely identified by the faulty switching element sets of the faulty paths in the detection phases. Furthermore, four, two or zero additional tests may be needed to differentiate $\phi\phi$ and -- as described in Theorem 4.3. Hence the number of tests needed is equal to at most eight. Q.E.D.

The single fault diagnosis scheme is good under the assumption that the diagnosis procedure can be repeated in a reasonably short period during which at most a single fault could possibly occur. However, it is well known that many physical faults of a single logical circuit component cannot be represented as a single fault.

4.3 Detection of Multiple Faults

Now we consider the detection problem for multiple faults. By a multiple fault, we mean the simultaneous occurence of any possible combination of single faults.

In the single-fault detection problem, we derive tests for every stuck-type fault at the link and functional state fault in the switching element. For the multiple fault case, the test set derived for detecting single faults may fail to indicate the existence of the fault because some faults may be masked by some other faults. We would show an example on the subnetwork shown in Fig. 4.1. Assume all the four switching elements in the subnetwork are in (S_5,S_5) functional state. Then the test vectors labelled on the input terminals will result in a correct response in the phase 1 test, although all the four switching elements are faulty. The faulty state which can mask a fault such that the fault becomes unobservable is called the masking faulty state. In valid state S_{10}, the masking faulty states are S_5, S_{12} and S_3, and in valid State S_5, the masking faulty states are S_{10}, S_{12} and S_3. The masking problem of the example shown in Fig. 4.1 can be solved by using four distinctive test vectors. Extending the solution to the whole network, we should use N distinctive test vectors for N terminals. The all-zero and all-one vectors should be excluded because these two vectors fail to test stuck-type faults at links. Hence, $1 + \log_2 N$ binary bits are needed to form the test vectors for the multiple fault. Two test phases, similar to the two for detecting single faults, are also needed for detecting multiple faults. Concluding the above discussion we have the following theorem.

Theorem 4.8: The number of tests for detecting multiple faults is equal to $2(1 + \log_2 N)$.

4.6 Summary

In this chapter, we have presented a fault model for the network in the class of multistage interconnection networks. Fault diagnosis procedures for the network constructed of switching elements with two valid states have been considered. A diagnosis method for single faults and a detection method for mutiple faults are developed. In the diagnosis procedures the control lines of the switching elements in the same stage can be grouped together and activated by the same control signal. The control line grouping of each stage is exactly the control scheme used in the flip network of STARAN [19]. Hence, the diagnosis procedures developed in this paper are good both for the indirect binary n-cube network and the flip network. Extension to the network constructed of switching elements with four valid states is feasible since the test sets of faults in switching elements with four valid states are the same as those we developed for switching elements with two valid states. The problem left is to design diagnosis procedures with minimal or nearly minimal number of tests.

The number of tests which is required under various conditions in the diagnosis procedures developed in this paper is summarized as follows. The number of tests for detecting single faults is equal to four and is independent of the network size. The number of tests for detecting multiple faults is equal to $2(1+\log_2 N)$, where N is the number of terminal links in one side of the network. The number of tests needed for determining the fault location and the fault type of a single fault depends on the fault type and/or the size of the network. The characteristics of single switching element faults are

Table 4.12 Characteristics of Single Faults

Cases		Fault Types	Faulty Output Pattern	No. of Fault Types	No. of Tests Needed For Determining the Fault Location	No. of Additional Tests For Determining the Fault Type
Case 1: One-Response Fault		$(S_3,S_5),(S_{12},S_5),(S_{10},S_3),(S_{10},S_{12})$	Table 4.4	4	$4+2\lfloor\log_2(\log_2 N)\rfloor$ or $4+2\lceil\log_2(\log_2 N)\rceil$	0
		$(S_2,S_5),(S_8,S_5),(S_{11},S_5),(S_{14},S_5),$ $(S_{10},S_1),(S_{10},S_4),(S_{10},S_7),(S_{10},S_{13})$	Table 4.4	8	$4+2\lfloor\log_2(\log_2 N)\rfloor$ or $4+2\lceil\log_2(\log_2 N)\rceil$	2
Case 2: Separated Two-Response Fault	Subcase A	$(S_{12},S_{12}),(S_{12},S_3),(S_3,S_{12}),(S_3,S_3)$	Table 4.8	4	4 or 8	0
	Subcase **B & C**	$(S_{12},S_{13}),(S_{12},S_7),(S_3,S_{13}),(S_3,S_7),$ $(S_{11},S_{12}),(S_{11},S_3),(S_{14},S_{12}),(S_{14},S_3),$ $(S_{12},S_4),(S_{12},S_1),(S_3,S_4),(S_3,S_1),$ $(S_2,S_{12}),(S_2,S_3),(S_8,S_{12}),(S_8,S_3)$	Table 4.9	16	4 or 8	2
	Subcase D & E	$(S_2,S_{13}),(S_2,S_7),(S_8,S_{13}),(S_8,S_7),$ $(S_{11},S_4),(S_{11},S_1),(S_{14},S_4),(S_{14},S_1),$ $(S_{11},S_{13}),(S_{11},S_7),(S_{14},S_{13}),(S_{14},S_7)$	Table 4.10	12	4,8 or 12	0 or 4
	Subcase **F**	$(S_2,S_4),(S_2,S_1),(S_8,S_4),(S_8,S_1)$	Table 4.11	4	(cannot be located at the single switching element level)	**(not a distinguishable from a link stuck fault)**
Case 3: Non-Separated Two-Response Fault		$\{S_0,S_1,S_4,S_5,S_6,S_7,S_9,S_{13},S_{15}\}\times\{S_5\};$ $\{S_{10}\}\times\{S_0,S_2,S_6,S_8,S_9,S_{10},S_{11},S_{14},S_{15}\}$		18	4	0 or 2
Case 4: Multiple-Response Fault		$\{S_0,S_1,S_4,S_5,S_6,S_7,S_9,S_{13},S_{15}\}\times\{S-S_5\};$ $\{S-S_{10}\}\times\{S_0,S_2,S_6,S_8,S_9,S_{10},S_{11},S_{14},S_{15}\}$		189	4	0, 2, or 4

summarized in Table 4.12. The minimum number of tests needed for determining the fault location and the fault type is equal to four and the maximum max(12, $6 + 2\lceil \log(\log N) \rceil$). For a network of size $N = 1024$ the maximum is equal to 14. There exist four switching element faults (Subcase F) which cannot be pinpointed at the single switching element level and those four are not distinguishable from the link stuck fault. This study provides specific information of fault characteristics for designing an easily diagnosable network.

CHAPTER 5

THE REVERSE-EXCHANGE INTERCONNECTION NETWORK

In many parallel processing architectures, an interconnection network is used to realize various permutations of data between the processors or between the processors and the memory modules. Due to the importance of the parallel processing, the design of cost effective interconnection networks is a crucial problem. In this concern some interconnection networks were proposed as described in Chapter 2. Among these networks the flip network was implemented in STARAN [19] and the shuffle-exchange network has been extensively investigated. The perfect shuffle permutation, on which the shuffle-exchange network is based, was studied by Golomb [70] and used by Pease [41] in the realization of the fast Fourier transformation. The shuffle-exchange network was presented by Stone [71] as an interconnection network between the cells of a dynamic memory. A generalization of Stone's network was been proposed by Lawrie [36] and extended by Lang [72] for the processor-memory interconnection in an array computer.

As far as these interconnection networks are concerned, the remaining problems include how to realize all permutations and how to develop efficient routing algorithms. The solutions may be approached in different ways. Our study on the addressing schemes results in a new interconnection network named the reverse-exchange network. In this chapter we will introduce this new network and investigate its usefulness to the remaining problems. Section 5.1 describes the reverse-exchange network. Some basic classes of permutations which are realizable by the reverse-exchange network are shown in Section 5.2. Routing algorithms which determine the control pattern by the permutation name are developed in Section 5.3 for those realizable permutations. In Section 5.4 we prove that the reverse - exchange network can realize all permutations in two passes. Both the construction and the routing scheme are provided. Some applications on the parallel processing are shown in Section 5.5.

5.1 The Reverse-Exchange Network

The network in the class of multistage interconnection networks, labelled by using the binary tree coding scheme, is actually a reverse - exchange network. In this section, we illustrate the reverse-exchange network using the baseline network.

The reverse-exchange network, shown in Fig. 5.1, which connects $N=2^n$ terminals on Side 1 to 2^n terminals on Side 2, is composed of n stages of 2 × 2 switching elements linked by a bit-reverse interconnection. Each 2 × 2 switching element can either send their inputs straight through (state 0) or exchange them (state 1). The bit-reverse interconnection is an interconnection when all switching elements in the network are in state 0, the positional relationship between Side 1 and Side 2 is in bit-reverse order. The interconnection is divided into n+1 levels. The leftmost and rightmost levels are identity interconnections. The interconnections between two adjacent stages are described by Eqs. (3.1) and (3.2).

The permutation function of the reverse-exchange network is accomplished by two components - the interconnection links and the switching elements. The interconnection links perform the bit-reverse permutation and the switching elements perform the exchange permutation. Assume the binary representation of integer X is $x_\ell x_{\ell-1} \cdots x_0$, where $\ell=n-1$. The interconnection link of level i, performs the following permutation:

$$R_i(x_\ell x_{\ell-1} \cdots x_0) = x_\ell \cdots x_{\ell-i+2} x_0 x_{\ell-i+1} \cdots x_1, \tag{5.1}$$

for $0 < i \leq \ell$. For i=0 and $\ell+1$,

$$R_i(x_\ell x_{\ell-1} \cdots x_0) = x_\ell x_{\ell-1} \cdots x_0. \tag{5.2}$$

By Eqs. (5.1) and (5.2) we have

$$R_{\ell+1}(R_\ell(R_{\ell-1} \cdots (R_0(x_\ell x_{\ell-1} \cdots x_0)) \cdots)) = x_0 x_1 \cdots x_\ell, \tag{5.3}$$

Eq. (5.3) implies that the overall interconnection links of the network perform the bit-reverse permutation. The exchange is performed by a switching element on two inputs named by adjacent numbers. The exchange

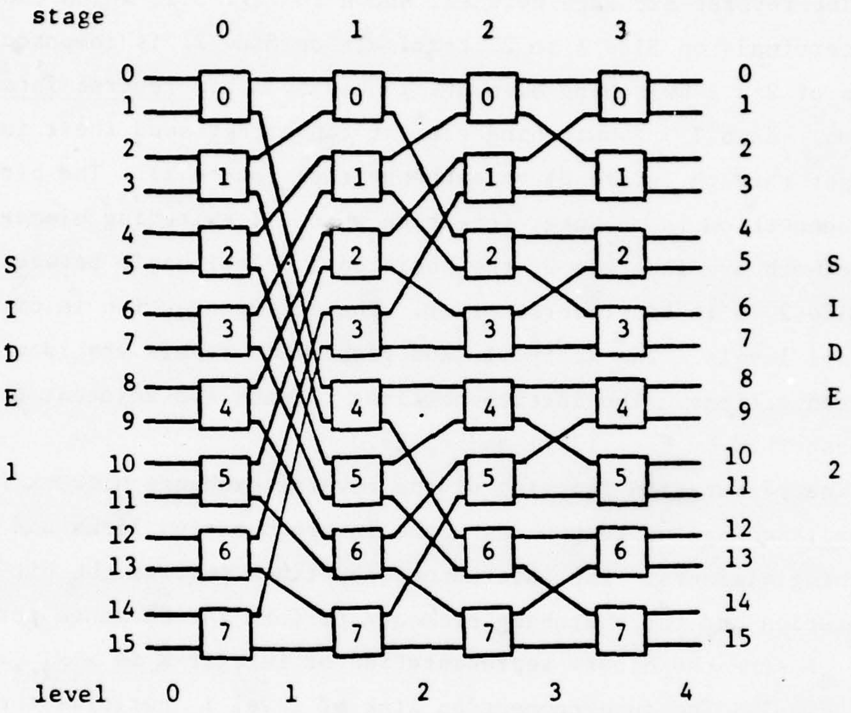


Fig. 5.1 Configuration of a reverse-exchange network.

permutation is defined as

$$E(x_\ell x_{\ell-1}\cdots x_0) = \begin{cases} x_\ell x_{\ell-1}\cdots x_0 & \text{if } c=0 \text{ ,} \\ x_\ell x_{\ell-1}\cdots \overline{x}_0 & \text{if } c=1 \text{ ,} \end{cases} \qquad (5.4)$$

where c is the control bit of the switching element and c=0 for state S_{10} and c=1 for state S_5. Since there are N/2 switching elements in a stage, there is a control vector associated with each stage. The notation of $C_i(j)$ and E_i are used to denote, respectively, the control bit for jth switching element in.stage i and the exchange permutation of stage i associated with control vector C_i.

The permutation of the reverse-exchange network realized is determined by the value of the control vector C_i's, $0 \le i \le \ell$. Assume X is permuted to P(X) by the network. Then

$$\begin{aligned} P(x_\ell x_{\ell-1}\cdots x_0) &= R_{\ell+1}(E_\ell(R_\ell(E_{\ell-1}\cdots(R_1(E_0(R_0(x_\ell x_{\ell-1}\cdots x_0))))\cdots))) \\ &= (e_0(x_0)\cdot e_1(x_1)\cdots e_{\ell-1}(x_{\ell-1})\cdot e_\ell(x_\ell)) \quad , \qquad (5.5) \end{aligned}$$

where $e_i(x_i)$, $0 \le i \le \ell$, is equal to x_i or $\overline{x}_i$ depending on the exchange performed by the associated switching element in stage i. Fig. 5.2 shows an example of a permutation realized on the network of size $N=2^3$ with control vectors as specified.

To have the reverse-exchange network perform the permutation, it is necessary to be able to derive the control vectors according to the permutation specifications. The homogeneous routing technique of the binary tree coding method shown in Chapter 3 are developed for the reverse-exchange network. Using this routing technique we can calculate the control vectors based on the source-destination pair of binary names. However, similar to the flip network and the shuffle - exchange network, not every permutation can be realized by the reverse - exchange network. The reverse-exchange network is capable of realizing $2^n \times 2^{n-1}$ permutations, which is considerably less than $2^n!$, the total number of possible permutations on $\{0,1,2,\ldots,2^n-1\}$. It is noted that the number of permutations which are useful in parallel computation is also much less than $2^n!$. The usefulness of the reverse-exchange network in parallel computations depends on how many permutations useful in parallel computations can be realized by the network in one pass or multiple passes.

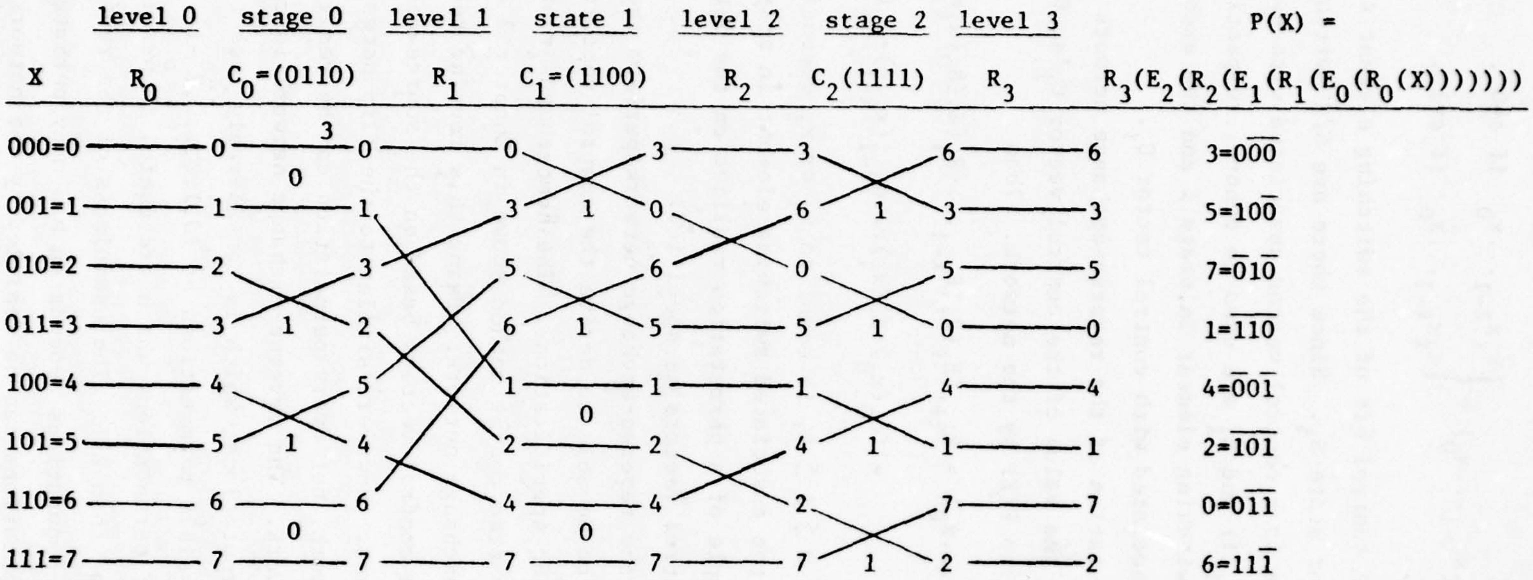


Fig. 5.2 A permutation realized by the reverse-exchange network of size 2^3.

5.2 Permutations Realizable by the Reverse-Exchange Network

The permutations realizable by the reverse-exchange network are a subset of $2^n!$ distinct permutations. In this section we will demonstrate some permutations in this subset, which are useful for the parallel processing.

Recall from Chapter 3 that given the source-destination pair of permutation request $P(A_j) = Z_j$, represented by $(A_j,Z_j) = (a_{j,\ell}a_{j,\ell-1}\dots a_{j,0}, z_{j,\ell}z_{j,\ell-1}\dots z_{j,0})$, stage m will switch source $a_{j,\ell}a_{j,\ell-1}\dots a_{j,0}$ to link $z_{j,\ell}z_{j,\ell-1}\dots z_{j,\ell-m+1}a_{j,\ell}a_{j,\ell-1}\dots a_{j,m+1}z_{j,\ell-m}$ in level m+1. A conflict occurs if some other source is also switched to this link. That is, for some pairs of permutation requests, say (A_j,Z_j) and (A_k,Z_k), and for some m, we have $a_{j,\ell}a_{j,\ell-1}\dots a_{j,m+1} = a_{k,\ell}a_{k,\ell-1}\dots a_{k,m+1}$ and $z_{j,\ell}z_{j,\ell-1}\dots z_{j,\ell-m} = z_{k,\ell}z_{k,\ell-1}\dots z_{k,\ell-m}$. If we define $A_{p,q} = a_{p,\ell}a_{p,\ell-1}\dots a_{p,\ell-q+1}$ and $Z_{p,q} = z_{p,\ell}z_{p,\ell-1}\dots z_{p,\ell-q+1}$, the conflict condition can be represented as $A_{j,\ell-m} = A_{k,\ell-m}$ and $Z_{j,m+1} = Z_{k,m+1}$ for permutation requests $p(A_j) = Z_j$ and $p(A_k) = Z_k$. Thus the following theorem defines the class of permutations which can be realized by the reverse-exchange network.

Theorem 5.1: Given a set of distinct permutation requests, $P_N = \{(A_i,Z_i) | 0 \le i < N\}$, P_N can be realized by the reverse-exchange network if and only if $A_j \ne A_k$ and $A_{j,\ell-m} = A_{k,\ell-m}$ implies $Z_{j,m+1} \ne Z_{k,m+1}$ for $j \ne k$, $0 \le j,k < N$ and $0 \le m \le \ell$.

Using Theorem 5.1 we will identify some of the permutations which can be realized by the reverse-exchange network.

Theorem 5.2: Define X_i^r to be the number whose binary representation is the reverse binary representation of X_i and define $P_N = \{(X_i,X_i^r) \mid 0 \le i < N)\}$ to be the bit-reverse permutation. Then P_N is realizable by the reverse-exchange network.

Proof: Assume $X_{j,\ell-i} = X_{k,\ell-i}$ for $0 \le i \le \ell-1$ where $j \ne k$. Since $X_j \ne X_k$ for $j \ne k$ we then obtain $X^r_{j,i+1} \ne X^r_{k,i+1}$ from the assumption. The proof immediately follows Theorem 5.1. Q.E.D.

<u>Theorem 5.3</u>: The permutation defined by $P_N = \{(X_i, aX_i^r) \mid 0 \leq i < N$ and a is an odd integer$\}$ is realizable by the reverse-exchange network.

<u>Proof</u>: Define $Y_i = aX_i^r$. We will prove that $X_{j,\ell-m} = X_{k,\ell-m}$ for $j \neq k$ and $0 \leq m < \ell$ implies $Y_{j,m+1} \neq Y_{k,m+1}$.

Since $X_{j,\ell-m} = X_{k,\ell-m}$ for $j \neq k$ and $0 \leq m < \ell$ implies $X^r_{j,m+1} \neq X^r_{k,m+1}$, hence, $x_{j,\ell}x_{j,\ell-1}\ldots x_{j,m+1} = x_{k,\ell}x_{k,\ell-1}\ldots x_{k,m+1}$ and $x_{j,m}x_{j,m-1}\ldots x_{j,0} \neq x_{k,m}x_{k,m-1}\ldots x_{k,0}$. Assume further that $x_{j,t} \neq x_{k,t}$ for some t, $0 \leq t \leq m$ and $x_{j,s} = x_{k,s}$ for $t < s \leq m$. Let $b_\ell b_{\ell-1}\ldots b_0$ be the binary representation of a where $b_0 = 1$ since a is odd. The products of $Y_j = aX^r_j$ and $Y_k = aX^r_k$ are shown as follows:

$$\begin{array}{llllll|llllll}
 & x_{j,0} & \cdot & \cdot & \cdot & x_{j,t} & x_{j,t+1} & \cdot & \cdot & & \cdot & x_{j,\ell} \\
\hline
\times & b_\ell & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & b_1 & b_0 \\
\hline
 & x_{j,0} & \cdot & \cdot & \cdot & x_{j,t} & x_{j,t+1} & \cdot & x_{j,\ell-2} & & x_{j,\ell-1} & x_{j,\ell} \\
 & b_1x_{j,1} & \cdot & \cdot & b_1x_{j,t} & b_1x_{j,t+1} & & \cdot & \cdot & & b_1x_{j,\ell-1} & b_1x_{j,\ell} \\
 & b_2x_{j,2} & \cdot & b_2x_{j,t} & b_2x_{j,t+1} & & \cdot & \cdot & \cdot & & b_2x_{j,\ell} & \\
+ & \cdot & \cdot & \cdot & \cdot & & \cdot & & & & & \\
\hline
 & y_{j,0} & \cdot & \cdot & y_{j,t-1} & y_{j,t} & y_{j,t+1} & \cdot & y_{j,\ell-2} & & y_{j,\ell-1} & y_{j,\ell} \\
\end{array}$$

and

$$\begin{array}{llllll|llllll}
 & x_{k,0} & \cdot & \cdot & \cdot & x_{k,t} & x_{k,t+1} & \cdot & \cdot & & \cdot & x_{k,\ell} \\
\hline
\times & b_\ell & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & b_1 & b_0 \\
\hline
 & x_{k,0} & \cdot & \cdot & \cdot & x_{k,t} & x_{k,t+1} & \cdot & x_{k,\ell-2} & & x_{k,\ell-1} & x_{k,\ell} \\
 & b_1x_{k,1} & \cdot & \cdot & b_1x_{k,t} & b_1x_{k,t+1} & & \cdot & \cdot & & b_1x_{k,\ell-1} & b_1x_{k,\ell} \\
 & b_2x_{k,2} & \cdot & b_2x_{k,t} & b_2x_{k,t+1} & & \cdot & \cdot & \cdot & & b_2x_{k,\ell} & \\
+ & \cdot & \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot & & & \\
\hline
 & y_{k,0} & \cdot & \cdot & y_{k,t-1} & y_{k,t} & y_{k,t+1} & \cdot & y_{k,\ell-2} & & y_{k,\ell-1} & y_{k,\ell} \\
\end{array}$$

The right sides of the dark lines in both cases are equivalent since

$X_{j,\ell-t} = X_{k,\ell-t}$. The above result shows that $Y_{j,m} = Y_{k,m}$ and $Y_{j,t} \neq Y_{k,t}$ if $X_{j,t} \neq X_{k,t}$ and $X_{j,m} = X_{k,m}$ for $t < m \leq \ell$. This result concludes that $X_{j,\ell-m} = X_{k,\ell-m}$ for $j \neq k$ and $0 \leq m < \ell$ implies $Y_{j,m+1} \neq Y_{k,m+1}$. Q.E.D.

<u>Theorem 5.4</u>: If $P_N = \{(A_i,Z_i) \mid 0 \leq i < N\}$ is realizable by the reverse-exchange network and b is an integer, then $P'_N = \{(A_i,Z_i+b) \mid 0 \leq i < N\}$ is also realizable by the network.

<u>Proof</u>: Let $Y_i = Z_i+b$. It is obvious that $Z_{j,m+1} \neq Z_{k,m+1}$ for $0 \leq m < \ell$ implies that $Y_{j,m+1} \neq Y_{k,m+1}$. Hence $A_{j,\ell-m} = A_{k,\ell-m}$ for $j \neq k$ and $0 \leq m < \ell$ implies $Z_{j,m+1} \neq Z_{k,m+1}$ and also $Y_{j,m+1} \neq Y_{k,m+1}$.
Q.E.D.

<u>Corollary 5.1</u>: The permutation defined by $P_N = \{(X_i, aX_i^r + b) \mid 0 \leq i < N$, a is an odd integer and b is an integer$\}$, is realizable by the reverse-exchange network.

<u>Corollary 5.2</u>: The permutation defined by $P_N = \{(X_i, T-X_i^r) \mid 0 \leq i < N$ and T is an integer$\}$ is realizable by the reverse-exchange network.

Corollaries 5.1 and 5.2 are consequences of Theorems 5.3 and 5.4.

<u>Theorem 5.5</u>: If $P_N = \{(A_i,Z_i) \mid 0 \leq i < N\}$ is realizable by the reverse-exchange network and k is an integer, then $P'_N = \{(A_i, Z_i \oplus k) \mid 0 \leq i < N$ and $\oplus$ is the bit-by-bit EXCLUSIVE OR$\}$ is also realizable by the network.

<u>Proof</u>: Define $Y_i = Z_i \oplus k$. It can be seen that $Y_{j,m} \neq Y_{k,m}$ if $Z_{j,m} \neq Z_{k,m}$ for $0 \leq m \leq \ell$. Since P_N is realizable by the reverse-exchange network, $A_{j,\ell-m} = A_{k,\ell-m}$ for $j \neq k$ and $0 \leq m < \ell$ implies $Z_{j,m+1} \neq Z_{k,m+1}$. Hence $A_{j,\ell-m} = A_{k,\ell-m}$ for $j \neq k$ and $0 \leq m < \ell$ also implies $Y_{j,m+1} \neq Y_{k,m+1}$. Q.E.D.

<u>Theorem 5.6</u>: The permutation defined by $P_N = \{(aX_i + b, X_i^r) \mid 0 \leq i < N$, a is an odd integer and b is an integer$\}$ is realizable by the reverse-exchange network.

<u>Proof</u>: Define $Y_i = aX_i + b$. If $X_{j,\ell-m} = X_{k,\ell-m}$, it is obvious

that $Y_{j,\ell-m'} = Y_{k,\ell-m'}$ for some $m' \geq m$. Since $X_{j,\ell-m} = X_{k,\ell-m}$ for $j \neq k$ and $0 \leq m < \ell$ implies $X^r_{j,m+1} \neq X^r_{k,m+1}$, $Y_{j,\ell-m'} = Y_{k,\ell-m'}$ for $j \neq k$ and $0 \leq m' < \ell$ also implies $X^r_{j,m'+1} \neq X^r_{k,m'+1}$. Q.E.D.

Theorem 5.7: Define the following binary representations:

$$X_i = (x_{i,\ell}x_{i,\ell-1}\ldots x_{i,j}x_{i,j-1}\ldots x_{i,0})$$

$$U = (x_{i,j-1}\ldots x_{i,0})$$

$$Y_i = (y_{i,0}\ldots y_{i,j-1}x_{i,j}\ldots x_{i,\ell-1}x_{i,\ell})$$

and

$$V = (y_{i,j-1}\ldots y_{i,0}).$$

Assume $V = U + k \bmod 2^j$, where k is an integer. Then the permutation defined by $P_N = \{(X_i,Y_i) \mid 0 \leq i < N\}$ is realizable by the reverse-exchange network.

Proof: By the definition, if $(x_{p,m}x_{p,m-1}\ldots x_{p,0}) \neq (x_{q,m}x_{q,m-1}\ldots x_{q,0})$, then $(y_{p,m}y_{p,m-1}\ldots y_{p,0}) \neq (y_{q,m}y_{q,m-1}\ldots y_{q,0})$ for either $m \leq j$ or $m > j$. Hence $X_{p,\ell-m} = X_{q,\ell-m}$ for $p \neq q$ and $0 \leq m < \ell$ implies $Y_{p,m+1} \neq Y_{q,m+1}$. Q.E.D.

Theorem 5.8: If $P_N = \{(A_i,Z_i) \mid 0 \leq i < N\}$ is realizable by the reverse-exchange network, then $P'_N = \{(Z_i,A_i) \mid 0 \leq i < N\}$ is also realizable by the network.

Proof: Since P_N is realizable by the reverse-exchange network, $a_{j,\ell}\ldots a_{j,m+1} = a_{k,\ell}\ldots a_{k,m+1}$ for $j \neq k$ and $0 \leq m < \ell$ implies $z_{j,\ell}\ldots z_{j,\ell-m} \neq z_{k,\ell}\ldots z_{k.\ell-m}$. By contradiction, assume that in the case of $z_{j,\ell}\ldots z_{j,q+1} = z_{k,\ell}\ldots z_{k,q+1}$ for $j \neq k$ and $0 \leq q < \ell$ we can have $a_{j,\ell}\ldots a_{j.\ell-q} = a_{k,\ell}\ldots a_{k.\ell-q}$. Then there exists $m = \ell-q-1$ such that $a_{j,\ell}\ldots a_{j,m+1} = a_{k,\ell}\ldots a_{k,m+1}$ and $z_{j,\ell}\ldots z_{j,\ell-m} = z_{k,\ell}\ldots z_{k,\ell-m}$. This contradicts the statement shown in the beginning of this proof. Hence $z_{j,\ell}\ldots z_{j,q+1} = z_{k,\ell}\ldots z_{k,q+1}$ for $j \neq k$ and $0 \leq q < \ell$ implies $a_{j,\ell}\ldots a_{j,\ell-q} \neq a_{k,\ell}\ldots a_{k,\ell-q}$. Q.E.D.

In this section we have presented some theorems which prove that some classes of permutations are realizable by the reverse-exchange network in one pass.

5.3 Controlling the Reverse-Exchange Network

The homogeneous routing procedure described in Chapter 3 already provides a simple routing mechanism. It employs the n-bit source tag and the n-bit destination tag to determine the valid state of the in-path switching elements. The mechanism also facilitates a conflict resolution scheme. This routing mechanism suggests a general routing procedure for all permutations. However there are at least two related reasons for us to pursue alternative routing algorithms. First, it would be quite expensive to implement the general routing procedure for a large network. Second, the permutations which are useful for the parallel processing and realizable by the reverse-exchange network can be classified into a limited number of classes. It is preferable to determine the control pattern by the name of the class of permutations rather than to consider the general routing procedure. In this section we will classify the permutations which are described in Section 5.2, look into the characteristics of each class, and present a routing scheme which determines the control pattern by the permutation name.

The permutations which are proven to be realizable by the reverse-exchange network in one pass are classified into the following categories.

1) $F_k^{(n)} (0 \le k < 2^n)$: $X \to X^r \oplus k$

$\tilde{F}_k^{(n)} (0 \le k < 2^n)$: $X^r \oplus k \to X$

2) $C_{j,k}^{(n)} (0 \le j,\ k < 2^n,\ j \text{ odd})$: $X \to jX^r + k$

$\tilde{C}_{j,k}^{(n)} (0 \le j,\ k < 2^n,\ j \text{ odd})$: $jX^r + k \to X$

3) $R_{j,k}^{(n)} (0 \le j,\ k < 2^n,\ j \text{ odd})$: $jX + k \to X^r$

$\tilde{R}_{j,k}^{(n)} (0 \le j,\ k < 2^n,\ j \text{ odd})$: $X^r \to jX + k$

4) $S_{q,k}^{(n)} (0 \le q \le n,\ 0 \le k < 2^q)$: cyclic shift of amplitude k
within each segment of size 2^q
as described in Theorem 5.7.

The control bit of a switching element is denoted either by 0 or by 1 depending on whether the desired function is the direct connection or the crossed connection. A N/2 or 2^{n-1}-bit vector is needed to control each stage. The generated bit pattern for the network can be structured as a cascaded matrix of column vectors, a binary tree, or a reverse binary tree. For a cascaded matrix of column vectors, n vectors of length 2^{n-1} are concatenated forward or backward and form the $2^{n-1} \times n$ control matrix. In case of the reverse binary tree the generated vector of stage i ($0 \leq i \leq n-1$) is split into 2^{n-i-1} vectors of length 2^i. These 2^i-bit vectors of stage i are shuffled into a 2^{n-1}-bit vector, column i of the control matrix. In the case of the binary tree, the generated vector of stage i is split into 2^i vectors of length 2^{n-i-1}. These 2^{n-i-1}-bit vectors of stage i are concatenated into 2^{n-1}-bit control columns of the control matrix.

Denote by $M^{(n)}(P) = (m_{ij})$, $0 \leq i \leq 2^{n-1} - 1$ and $0 \leq j \leq n-1$, the control pattern of $2^{n-1} \times n$ matrix associated with a permutation P and by $K^{(n)}(P)$ the generated bit pattern of n columns according to recursive formulas to be demonstrated. Denote also by $v^{(n-j)}(b)$ the 2^{n-j}-bit vector whose components are all equal to b. A binary tree whose root is a vector v and whose upper subtree and lower subtree are K_u and K_v, respectively, is denoted by $[v;\ K_u,K_v\ ]$. Similarly, the reverse binary tree is denoted by $[\ K_u,K_v\ ;v]$. The cascaded matrix whose left part and right part are L and R, respectively, is denoted by [L;R].

Let k be a positive integer and denote, respectively, by k' and k_1, its quotient and its remainder in the division by 2, i.e., $k = 2k' + k_1$.

Algorithm 1(a): If $n \geq 2$,then

$$K^{(n)}(F^n_k) = [K^{(n-1)}(F^n_{k'});\ v^{(n-1)}(k_1)].$$

If n = 1, then

$$K^{(1)}(F^n_{k'}) = [v^{(n-1)}(k')].$$

Example: Assume

$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 7 & 1 & 5 & 2 & 6 & 0 & 4 \end{pmatrix}.$$

P can be described by

$$F_3^{(3)}: \ x \to x^r \oplus 3$$

According to Algorithm 1(a), we have

$$K^{(3)}(F_3^{(3)}) = [v^{(2)}(0);v^{(2)}(1);v^{(2)}(1)].$$

Hence,

$$M^{(3)}(P) = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

The setting of the network is illustrated in Fig. 5.3.

Algorithm 1(b): If $n \geq 2$, then

$$K^{(n)}(\tilde{F}_k^{(n)}) = [v^{(n-1)}(k_1);K^{(n-1)}(\tilde{F}_{k'}^{(n)})].$$

If $n = 1$, then

$$K^{(1)}(\tilde{F}_{k'}^{(n)}) = [\ v^{(n-1)}(k')].$$

Example: Assume

$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 5 & 3 & 7 & 0 & 4 & 2 & 6 \end{pmatrix}.$$

P can be described by

$$\tilde{F}_4^{(3)}: \ x^r \oplus 4 \to x.$$

According to Algorithm 1(b), we have

$$K^{(3)}(\tilde{F}_4^{(3)}) = [v^{(2)}(0);v^{(2)}(0);v^{(2)}(1)].$$

Hence

$$M^{(3)}(P) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

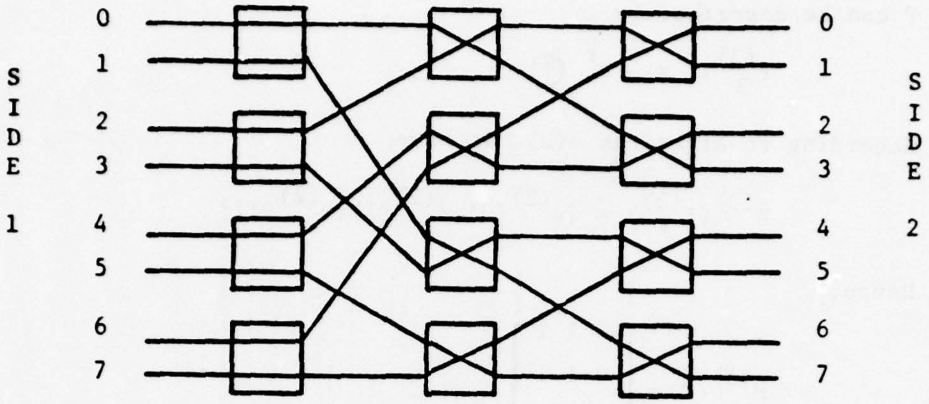


Fig. 5.3 Setting for $F_3^{(3)}$.

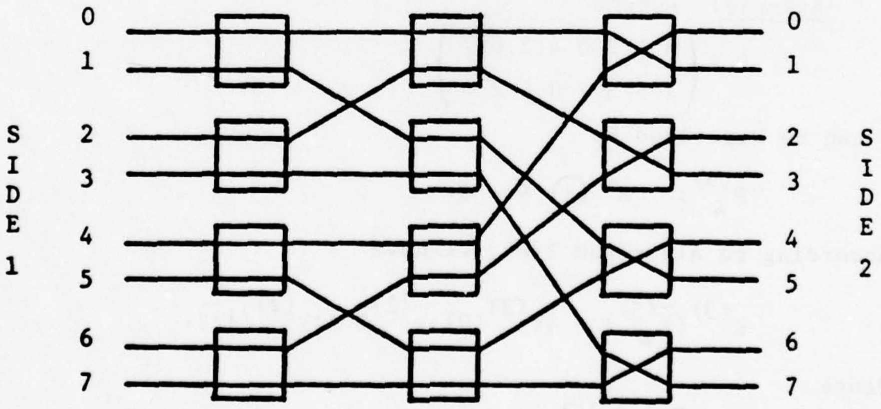


Fig. 5.4 Setting for $\tilde{F}_4^{(3)}$.

The setting of the network is illustrated in Fig. 5.4.

Algorithm 2(a): Let $j = 2j' + 1$. If $n \geq 2$, then

$$K^{(n)}(C^{(n)}_{j,k}) = \begin{cases} [K^{(n-1)}(C^{(n-1)}_{j,(j'+1)k_1+k'}),K^{(n-1)}(C^{(n-1)}_{j,(1-k_1)j'+k'}); \\ \qquad\qquad v^{(n-1)}(k_1)], \text{ if } k_1 = 0; \\ [K^{(n-1)}(C^{(n-1)}_{j,(1-k_1)j'+k'}),K^{(n-1)}(C^{(n-1)}_{j,(j'+1)k_1+k'}); \\ \qquad\qquad v^{(n-1)}(k_1)], \text{ if } k_1 = 1. \end{cases}$$

If $n = 1$, then

$$K^{(1)}(C^{(1)}_{j,k}) = [k].$$

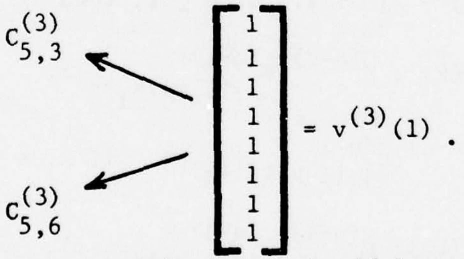
Example: Assume

$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 7 & 15 & 11 & 3 & 1 & 9 & 5 & 13 & 12 & 4 & 0 & 8 & 6 & 14 & 10 & 2 \end{pmatrix}.$$

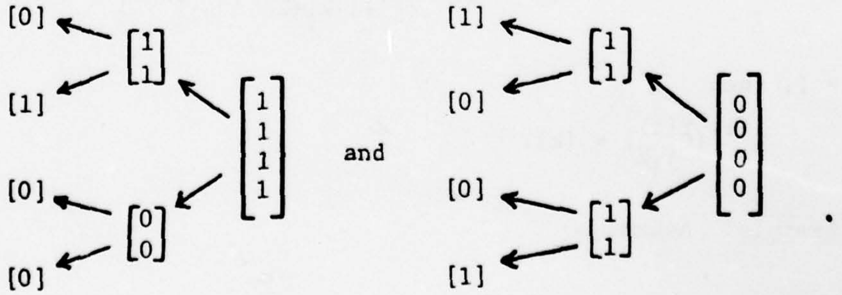
P can be described by

$$C^{(4)}_{5,7} : x \rightarrow 5x^r + 7.$$

The first application of Algorithm 2(a) results in



Similarly, we can apply Algorithm 2(a) to $C^{(3)}_{5,3}$ and $C^{(3)}_{5,6}$, and obtain, respectively, the reverse binary trees:



Thus we have

$$K^{(4)}(C^{(4)}_{5,7}) = \left[\begin{matrix} \begin{matrix} {[0]} \\ {[1]} \\ {[0]} \\ {[0]} \\ {[1]} \\ {[0]} \\ {[0]} \\ {[1]} \end{matrix} & \begin{matrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{matrix} & \begin{matrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix} & \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \end{matrix}\right] .$$

Shuffling the subvectors in each column of $K^{(4)}(C^{(4)}_{5,7})$,we can obtain

$$M^{(4)}(P) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} .$$

The setting of the network is illustrated in Fig. 5.5.

Algorithm 2(b): Let $j = 2j' + 1$. If $n \geq 2$, then

$$K^{(n)}(\tilde{C}^{(n)}_{j,k}) = \begin{cases} [v^{(n-1)}(k_1);\ K^{(n-1)}(\tilde{C}^{(n-1)}_{j,(j'+1)k_1+k'}), \\ \qquad K^{(n-1)}(\tilde{C}^{(n-1)}_{j,(1-k_1)j'+k'})], & \text{if } k_1 = 0; \\ [v^{(n-1)}(k_1);\ K^{(n-1)}(\tilde{C}^{(n-1)}_{j,(1-k_1)j'+k'}), \\ \qquad K^{(n-1)}(\tilde{C}^{(n-1)}_{j,(j'+1)k_1+k'})], & \text{if } k_1 = 1. \end{cases}$$

If $n = 1$, then

$$K^{(1)}(\tilde{C}^{(1)}_{j,k}) = [k].$$

Example: Assume

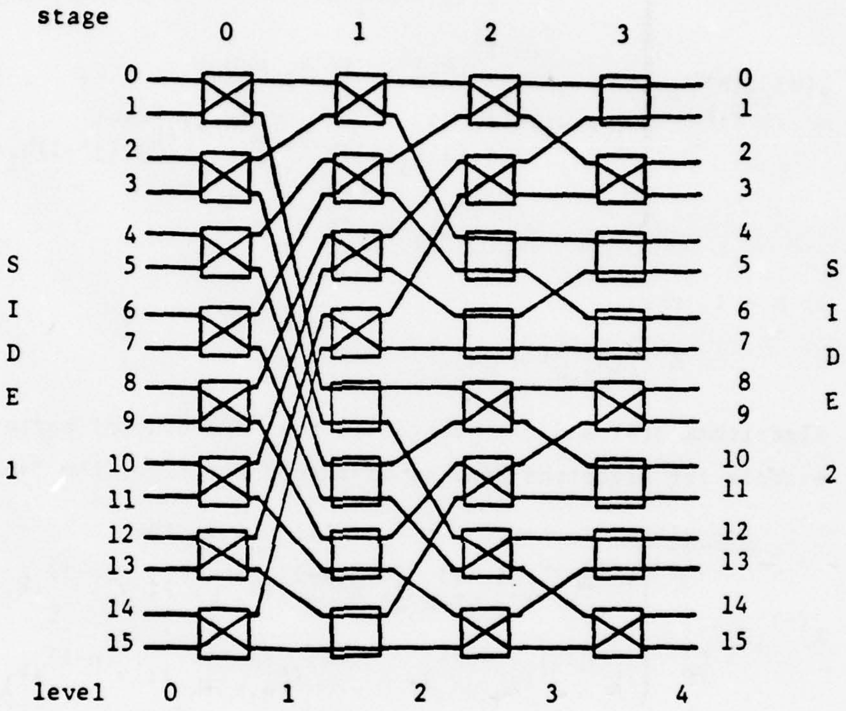


Fig. 5.6 Setting for $\tilde{C}^{(4)}_{5,7}$.

If $n = 1$, then

$$K^{(1)}(R^{(1)}_{j,k}) = [k].$$

Algorithms 3(a) and 2(b) result in the same control pattern. The example for Algorithm 2(b) is also good for Algorithm 3(a).

<u>Algorithm 3(b)</u>: Let $j = 2j' + 1$. If $n \geq 2$, then

$$K^{(n)}(\tilde{R}^{(n)}_{j,k}) = \begin{cases} [K^{(n-1)}(\tilde{R}^{(n-1)}_{j,(j'+1)k_1+k'}),\ K^{(n-1)}(\tilde{R}^{(n-1)}_{j,(1-k_1)j'+k'}); \\ \qquad v^{(n-1)}(k_1)], \text{ if } k_1 = 0 ; \\ [K^{(n-1)}(\tilde{R}^{(n-1)}_{j,(1-k_1)j'+k'}),\ K^{(n-1)}(\tilde{R}^{(n-1)}_{j,(j'+1)k_1+k'}); \\ \qquad v^{(n-1)}(k_1)], \text{ if } k_1 = 1. \end{cases}$$

If $n = 1$, then

$$K^{(1)}(\tilde{R}^{(1)}_{j,k}) = [k].$$

Algorithms 3(b) and 2(a) result in the same control pattern. The example for Algorithm 2(a) is also good for Algorithm 3(b).

<u>Algorithm 4</u>: Let $0 \leq q \leq n$. If $n \geq 2$, then

$$K^{(n)}(S^{(n)}_{q,k}) = \begin{cases} [K^{(n-1)}(S^{(n-1)}_{q,k'+k_1}),\ K^{(n-1)}(S^{(n-1)}_{q,k'});\ v^{(n-1)}(k_1)], \text{ if } k_1 = 0; \\ [K^{(n-1)}(S^{(n-1)}_{q,k'}),\ K^{(n-1)}(S^{(n-1)}_{q,k'+k_1});\ v^{(n-1)}(k_1)], \text{ if } k_1 = 1, \end{cases}$$

where $K^{(n-q)}(S^{(n-q)}_{q,k}) = K^{(n-q)}(C^{(n-q)}_{1,0})$. If $q = n$ (equivalent to Algorithm 2(a) for $j = 1$), then

$$K^{(1)}(S^{(1)}_{n,k}) = [k].$$

<u>Example</u>: Assume

$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 3 & 11 & 7 & 15 & 1 & 9 & 5 & 13 & 0 & 8 & 4 & 12 & 2 & 10 & 0 & 14 \end{pmatrix}.$$

P can be described by $S^{(4)}_{2,3}$ which is a cyclic shift of amplitude 3 within each segment of size 2^2 as shown in Fig. 5.7.

The first application of Algorithm 4 results in

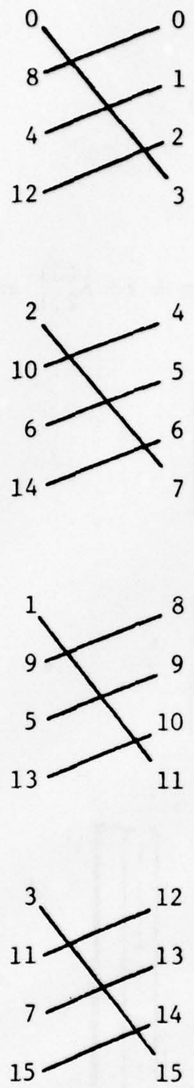


Fig. 5.7 Permutation of $S_{2,3}^{(4)}$.

$$\begin{tikzcd}
S^{(3)}_{2,1} & \\
 & \begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} = v^{(3)}(1)\ . \arrow[ul] \arrow[dl] \\
S^{(3)}_{2,2} &
\end{tikzcd}$$

Similarly, we can apply Algorithm 4 to $S^{(3)}_{2,1}$ and $S^{(3)}_{2,2}$ and obtain, respectively,

$$\begin{tikzcd}
C^{(2)}_{1,0} = S^{(2)}_{2,0} & \\
 & \begin{bmatrix}1\\1\\1\\1\end{bmatrix} \arrow[ul] \arrow[dl] \\
C^{(2)}_{1,0} = S^{(2)}_{2,1} &
\end{tikzcd}$$

and

$$\begin{tikzcd}
C^{2}_{1,0} = S^{2}_{2,1} & \\
 & \begin{bmatrix}0\\0\\0\\0\end{bmatrix} \arrow[ul] \arrow[dl] \\
C^{2}_{1,0} = S^{2}_{2,1} &
\end{tikzcd}$$

Hence we obtain

$$K^{(4)}(S^{(4)}_{2,3}) = \begin{bmatrix} \begin{matrix} {[0]} \\ {[0]} \\ {[0]} \\ {[0]} \\ {[0]} \\ {[0]} \\ {[0]} \\ {[0]} \end{matrix} & \begin{matrix} \begin{bmatrix}0\\0\end{bmatrix} \\ \begin{bmatrix}0\\0\end{bmatrix} \\ \begin{bmatrix}0\\0\end{bmatrix} \\ \begin{bmatrix}0\\0\end{bmatrix} \end{matrix} & \begin{matrix} \begin{bmatrix}1\\1\\1\\1\end{bmatrix} \\ \begin{bmatrix}0\\0\\0\\0\end{bmatrix} \end{matrix} & \begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} \end{bmatrix}$$

Consequently,

$$M^{(4)}(P) = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The setting of the network is illustrated in Fig. 5.8.

5.4 Realization of Arbitrary Permutations

In this section we will first show that all permutations can be realized by the reverse-exchange network in two passes. Next, we consider the routing scheme for this two-pass construction.

A. Two-pass permutations

The fact that the Benes binary network can realize all permutations between its inputs and outputs follows the result of Slepian-Dupid theorem. Using the above fact, we will prove that the reverse-exchange network can realize all permutations in two passes.

Theorem 5.9: The reverse-exchange network can realize all permutations in two passes.

Proof: The theorem will be proven by showing that the functions of the Benes binary network can be simulated by the baseline network in two passes. An example construction for a two-pass implementation using a baseline network is shown in Fig. 5.9. The input data are fed in on Side 1. The output data of the first pass are stored in the shift register files on Side 2. In the second pass, the data in the register files are fed back to the input lines on Side 1 and the final results are again stored in the register files.

The two-pass construction is equivalent to the implementation of cascading two baseline networks. An example, Fig. 5.10, shows the equivalent construction of the two-pass construction shown in Fig. 5.9. The two baseline networks shown in Fig. 5.10 are labelled with logical names.

According to the previous result we can obtain a reverse baseline network via properly permuting the switching elements and its related links of the baseline network. For switching elements in the reverse baseline network, the mapping, γ_i, from physical names, $(p_\ell p_{\ell-1} \cdots p_1)_i$ to logical names $(b_\ell b_{\ell-1} \cdots b_1)_i$ is shown in Chapter 3. The mapping, γ_i^{-1}, from logical names to

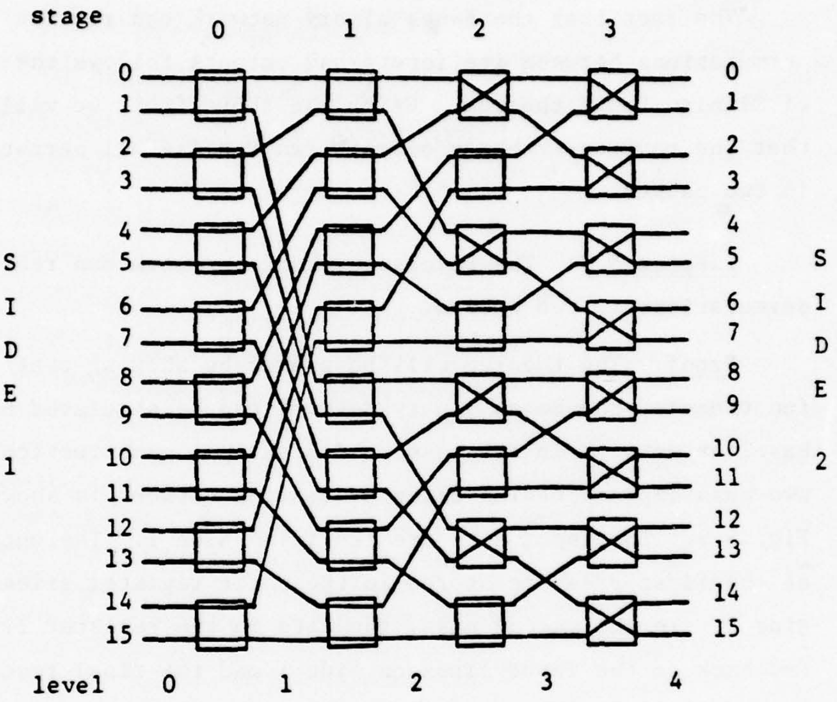


Fig. 5.8 Setting for $S^{(4)}_{2,3}$.

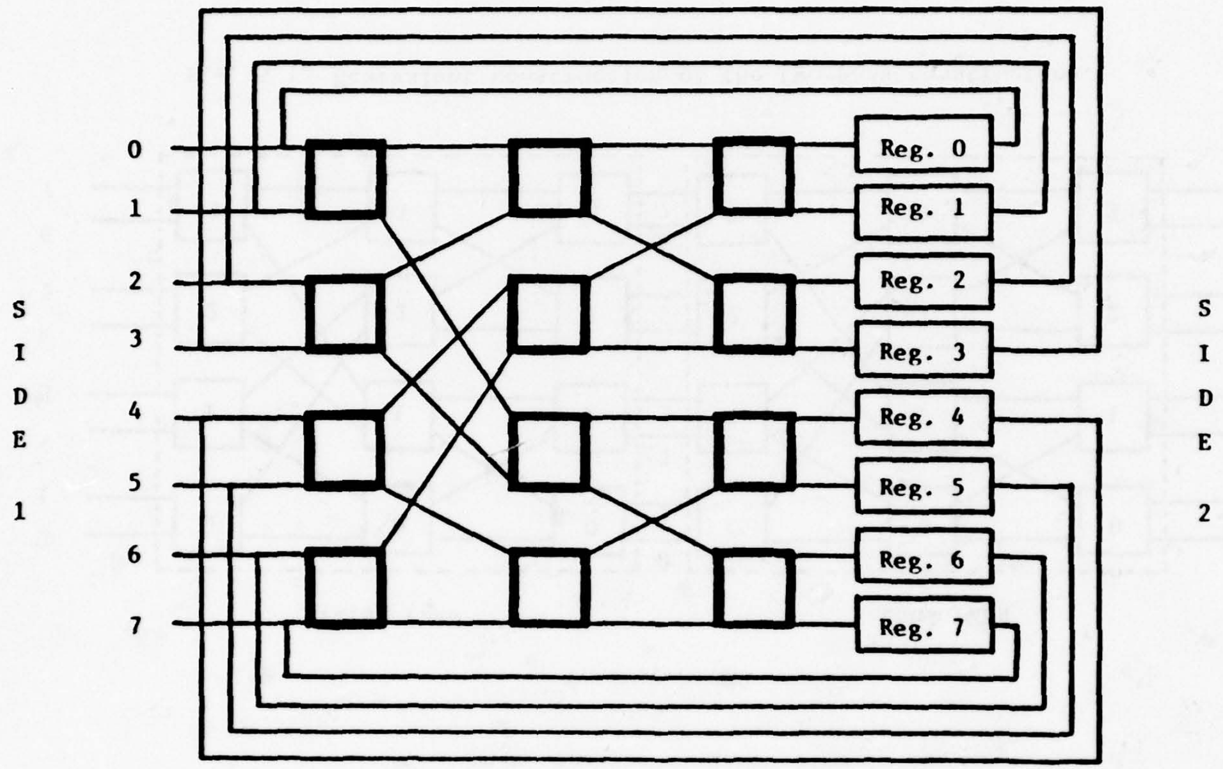


Fig. 5.9 Two-pass construction for a reverse-exchange network.

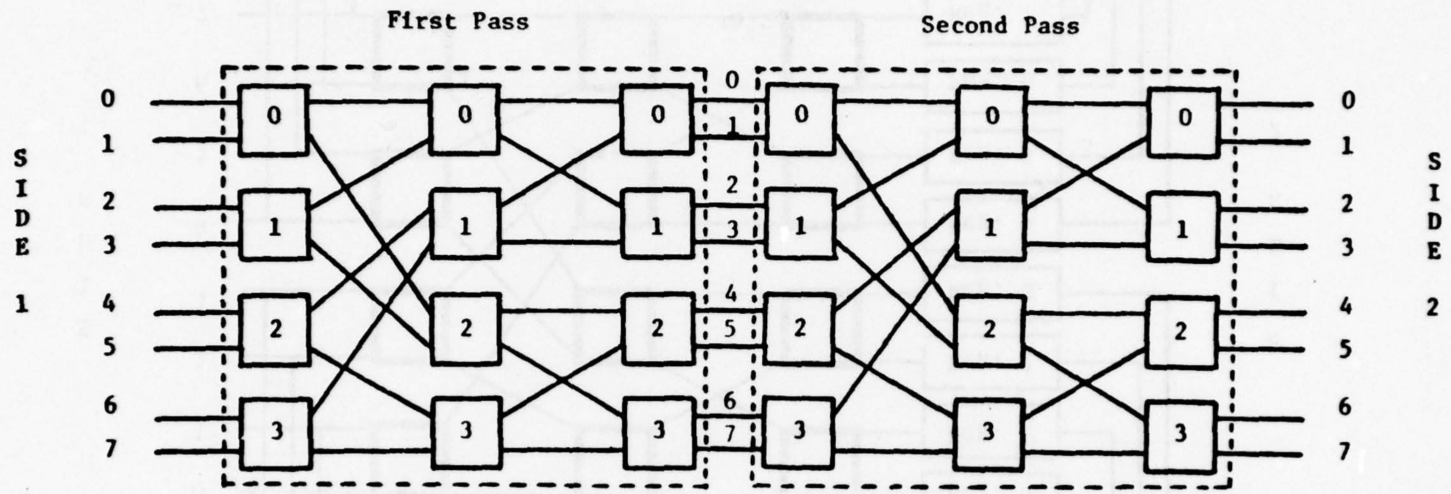


Fig. 5.10 Equivalent construction of the two-pass construction.

physical names is shown in the following:

$$\gamma_i^{-1}[(b_\ell b_{\ell-1} \ldots b_1)_i] = (b_{\ell-i} \ldots b_1\ b_\ell \ldots b_{\ell-i+1})_i, \qquad (5.6)$$

for $0 \le i \le \ell$. If we rearrange the switching elements and its related links of the baseline network for the second pass in the equivalent construction in ascending order of the physical names, which are obtained by applying γ_i^{-1} of Eq. (5.6) on the logical names, we can obtain a construction which is formed by connecting a baseline network to a reverse baseline network end by end. An example is shown in Fig. 5.11. The labellings shown in Fig. 5.11 are the logical names. Now, by setting the switching elements in the first stage of the reverse baseline network in the equivalent construction on the state of the direct connection, we then show that the equivalent construction functions exactly as a Benes binary network. The setting is shown in Fig. 5.12 for the example. It can be seen that the construction shown in Fig. 5.12 is equivalent to that shown in Fig. 5.13. Q.E.D.

B. Routing scheme

Several authors have proposed algorithms which compute control patterns for the Benes binary network for any one-to-one permutation assignment. Among these are the scheme by Opferman and Tsao-Wu [29] and the looping procedure by Anderson [73]. Although these two algorithms are good for any permutation assignment on the Benes binary network, they both need memory storage for implementing the algorithm and the computing time needed is in the order of $(N/2)\log_2(N/2)$. However, Lenfant [74] claimed that these algorithms are both time-consuming and space-consuming. In order to meet the time constraints arising from the use of a Benes binary network as the alignment network, Lenfant proposed a routing algorithm for frequently used permutations which are classified into five families. For each family the routing algorithm can control the two-state switches on the fly as the vector of data passes through the network.

These three algorithms can also be used in our two-pass construction which can realize all permutations as the Benes

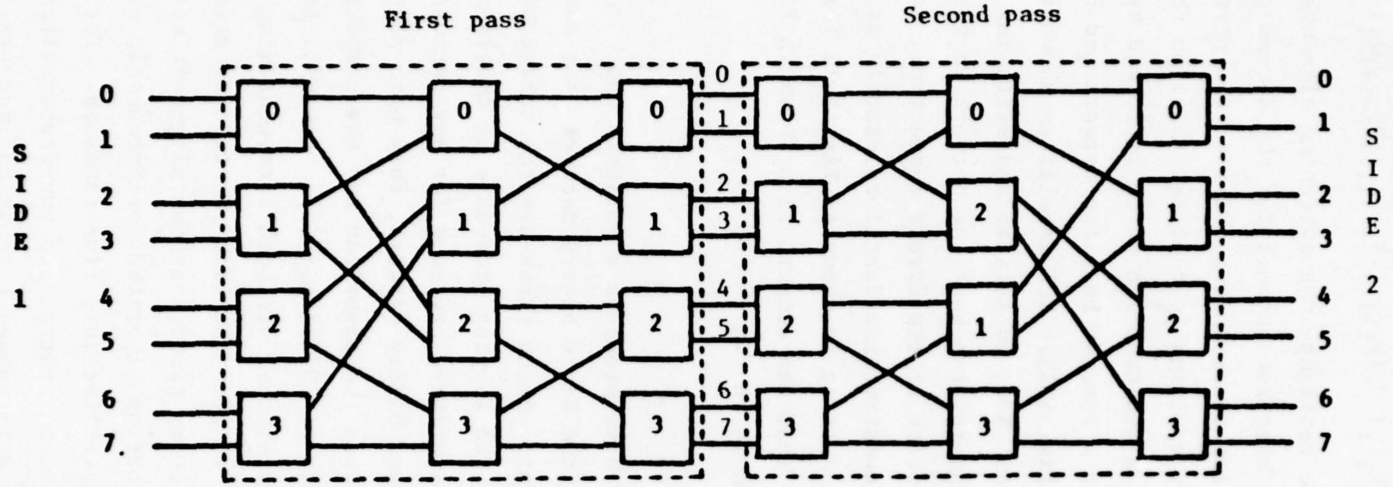


Fig. 5.11 Another structure of the equivalent construction.

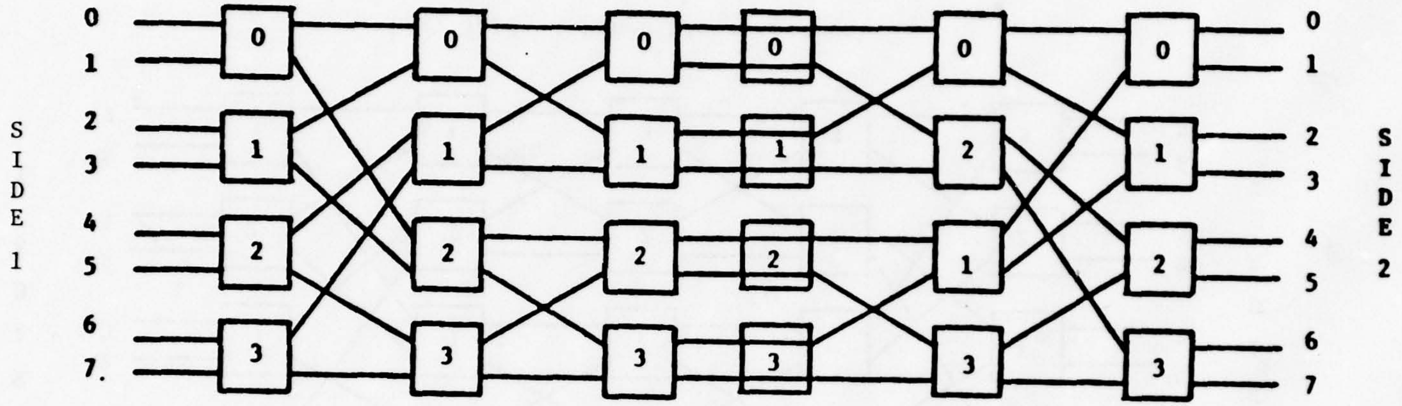


Fig. 5.12 Confinement of some switching elements in the second pass.

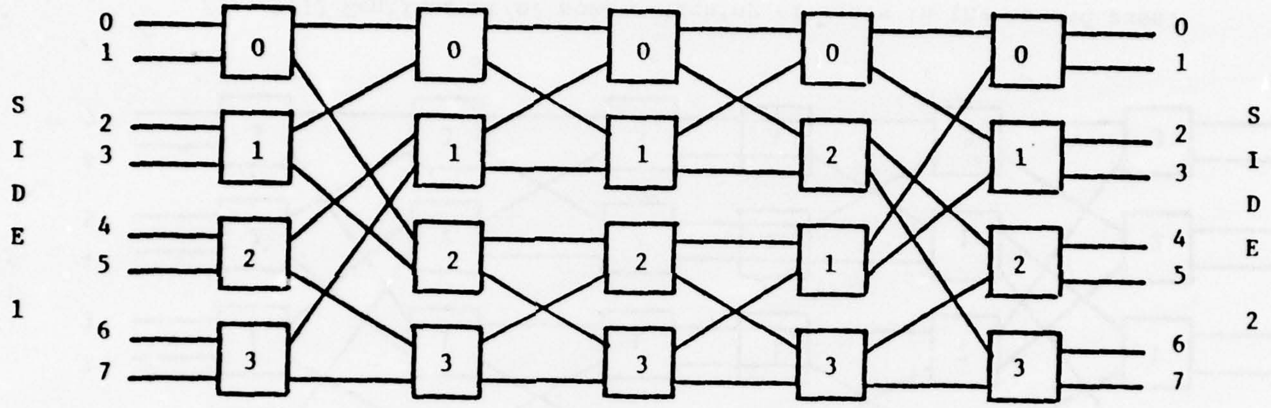


Fig. 5.13 A Benes binary network.

binary network does. However, the computed control pattern should properly be permuted before it can be applied to the reverse-exchange network. As shown in Theorem 5.9, the leftmost n stages of the Benes binary network are one-to-one correspondent to the reverse-exchange network of the first pass from left to right, and the rightmost n-1 stages of the Benes binary network are one-to-one correspondent to the reverse-exchange network of the second pass from right to left. The switching elements in the left most stage of the reverse-network of the second pass are refined to be in the valid state of the direct connection. Assume that signals 0 and 1 represent the valid states for the direct and the crossed connection. We can represent the control pattern of the Benes network and the reverse-exchange network by the following matrices:

1. Benes binary network

$$B = \begin{bmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,2n-2} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,2n-1} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ b_{\frac{N}{2},0} & b_{\frac{N}{2},0} & \cdots & b_{\frac{N}{2},2n-2} \end{bmatrix},$$

2. Reverse-exchange network of the first pass

$$R_1 = \begin{bmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,n-1} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ b_{\frac{N}{2},0} & b_{\frac{N}{2},1} & \cdots & b_{\frac{N}{2},n-1} \end{bmatrix},$$

3. Reverse-exchange network of the second pass

$$R_2 = \begin{bmatrix} 0 & a_{0,1} & \cdots & a_{0,n-1} \\ 0 & a_{1,1} & \cdots & a_{1,n-1} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & a_{\frac{N}{2},1} & \cdots & a_{\frac{N}{2},n-1} \end{bmatrix} .$$

Hence, given matrix B, we can immediately obtain matrix R_1 and derive matrix R_2 by performing the following permutation according to Eq. (5.6):

$$a_{j,i} = b_{k,n+i-1} , \qquad (5.7)$$

where $j = r_i^{-1}(k)$ and $1 \leq i \leq n-1$.

Example: Assume the following control pattern is computed by using one of the algorithms [29,72,73] for the 8 x 8 Benes binary network:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} .$$

The setting of the Benes binary network is illustrated in Fig. 5.14. We can immediately obtain

$$R_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} .$$

Since, according to Eq. (5.7), we have

$$\begin{cases} a_{0,1} = b_{0,4} \\ a_{1,1} = b_{2,4} \\ a_{2,1} = b_{1,4} \\ a_{3,1} = b_{3,4} \end{cases} ,$$

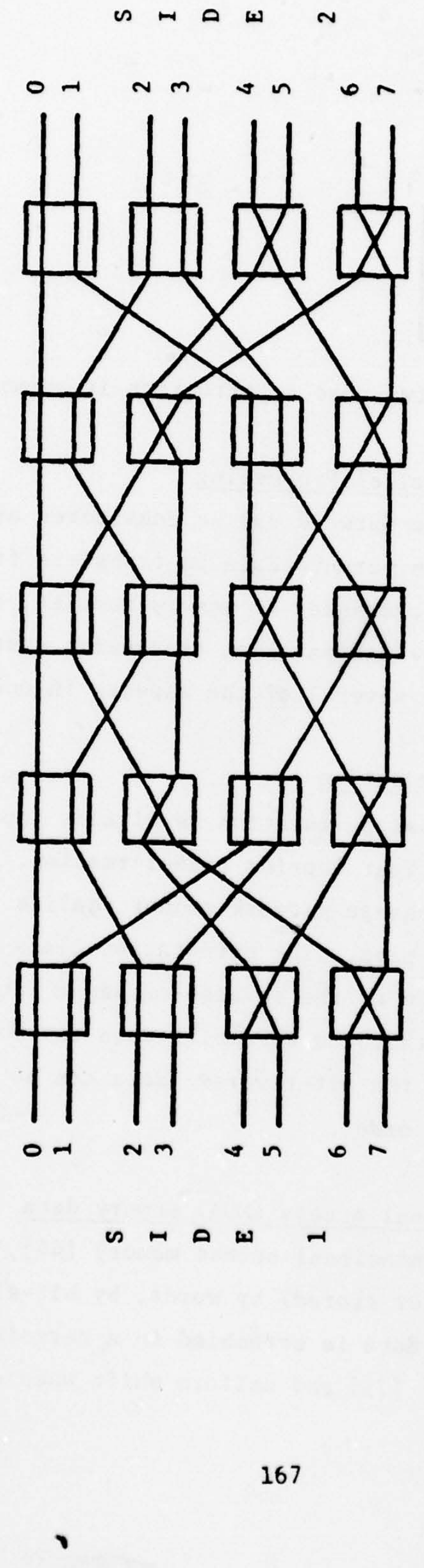


Fig. 5.14 A setting of the Benes binary network.

and

$$\begin{cases} a_{0,2} = b_{0,5} \\ a_{1,2} = b_{1,5} \\ a_{2,2} = b_{2,5} \\ a_{3,2} = b_{3,5} \end{cases},$$

we can obtain

$$R_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

The setting of the two-pass construction is shown in Fig. 5.15.

5.5 Applications on Parallel Processing

The reverse-exchange network can be considered as an interconnection network which can permute data on transfers from memory to processor modules, from processor to memory modules, and from processor to processor modules in a parallel processing system. In this section we will consider several of the aspects in one pass.

A. Bit-reverse permutation

The bit-reversal permutation is vitally important to the computation of the fast Fourier transformation. The flip network and the shuffle exchange network cannot realize the bit-reverse permutation in one pass. The permutation class of $R^{(n)}_{j,k}$ and $\tilde{R}^{(n)}_{j,k}$ which are realizable by the reverse-exchange network in one pass clearly indicates that the scrambled data can be aligned in bit-reverse order and the bit-reverse data can be restored in the original scrambled order.

B. Multi-dimensional access (MDA) memory data

In a multi-dimensional access memory [44], data can be accessed (fetched or stored) by words, by bit-slices, by byte-slices, etc. MDA data is scrambled in a certain way such as p-ordered scramble [75] and uniform shift when stored in memory

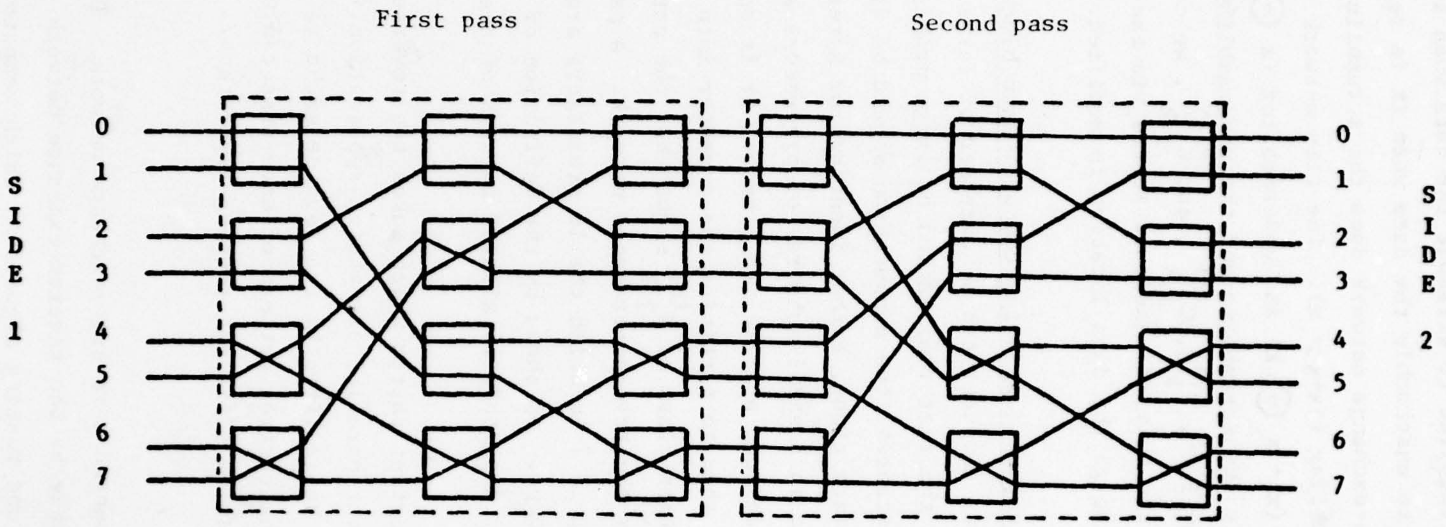


Fig. 5.15 A setting of the equivalent two-pass construction.

so that it can be accessed in various ways. A scramble/unscramble network is required to scramble the data when it is stored into memory and to unscramble the data when it is read from memory. The shuffle-exchange network does the scrambling ($x \to jx+k$) and the unscrambling ($jx+k \to x$). The flip network also does the scrambling ($x \to x \oplus k$) and unscrambling ($x \oplus k \to x$). However, if we modify the scrambling and the unscrambling a little bit as described by $F_k^{(n)}$, $\tilde{F}_k^{(n)}$, $C_{j,k}^{(n)}$, and $\tilde{C}_{j,k}^{(n)}$, we can achieve the same purpose of the multi-dimensional access via the reverse-exchange network although the data location is modified.

C. Partitioning of an array computer into blocks of 2^q processors

An array computer can be composed of large numbers of processors for the fast realization of large problems. However, in some circumstances, the computation should be divided into subgroups and each group, either identical or heterogeneous, can be performed in a small subarray of processors and achieve the efficiency through parallelism. Hence, it is convenient in these cases to be able to partition the computer into various subarrays. The permutation class of $S_{j,k}^{(n)}$ shows that the partition can be supported by the reverse-exchange network. A partition has been shown in Fig. 5.7 in which the 16 processors are partitioned into 2^q ($q=2$) groups. As shown by the definition of $S_{j,k}^{(n)}$, the cyclic shift of any amplitude is allowed in each of the subarrays.

Another important factor which makes the reverse-exchange network more favored to the parallel processing application is that all permutations can be realized by the reverse-exchange network in just two passes and the control pattern of each pass can easily be obtained by using the existing efficient routing algorithms.

5.6 Summary

We have presented a reverse-exchange network. The permutations which are realizable by the reverse-exchange network are classified into four groups and routing algorithms which compute the control patterns according to the permutation group names are developed.

It is also proven that all permutations can be realized by the reverse-exchange network in two passes. Both the construction and the routing algorithm are provided. The network is shown to be useful for the bit-reversal permutation, the multi-dimensional access memory and partitioning the array computer. Overall, the reverse-exchange network is a powerful interconnection network for the parallel processing system.

CHAPTER 6

LOGIC PARTITIONING OF MULTISTAGE INTERCONNECTION NETWORKS FOR LSI IMPLEMENTATION

Recent proposals on computer architecture consider computer systems with as many as 2^{14} to 2^{16} processors. The implementation of the interconnection networks in such large systems is a critical problem for the designers. The need of having a cost-effective LSI implementation of the interconnection network is obvious. However, there is very limited research activity on the issue of LSI implementation. In this Chapter we will tackle some problems on the LSI implementation of the baseline network. The results are also good for other equivalent networks.

For a cost-effective LSI implementation, the minimization of the number of modular types is of prime importance. Hence it would be a good criterion to partition the network into functionally and physically equivalent modules so that the hardware and the software can modularly be developed. However, there also exist some limitations on LSI technology. The maximum number of gates and pins allowed in an LSI chip are frequently used to describe the limitations. Also, the gate-to-pin ratio in the actual implementation has been used to measure the cost effectiveness. The problem of the LSI implementation here is then, given the maximum allowable number of gates and pins in an LSI chip, to implement the network with the minimum number of modular types and the maximum gate-to-pin ratio.

This study generates a generalized partition formula for the LSI implementation, a measurement on the cost-effectiveness and a scheme for interconnecting circuit chips. Section 6.1 first illustrates some examples for the logic partitioning and then shows a general partition formula. In Section 6.2 we manipulate the general partition formula for minimizing the number of modular types and in Section 6.3 we count pin numbers in the implementation and provide a measurement on the cost effectiveness. The scheme of interconnecting circuit chips in an implementation is developed in Section 6.4.

6.1 Partitioning

Assume the network of size $N=2^n$ cannot be implemented in a single circuit chip because of the pin or gate limitation. Partitioning the network into composite subnetworks then becomes a necessary design step. We will illustrate the partitioning on the example network shown in Fig. 3.3 and extend the partitioning to the general case. Fig. 6.1 shows a partition. As shown in Fig. 6.1(a), the network can be implemented by a subnetwork shown in Fig. 6.1(b), which has four switching elements and 16 pins. Another partition is shown in Fig. 6.2. In Fig. 6.2(a) we can see that the first and the last two stages are implemented, respectively, by four subnetworks of size 2^2. The subnetwork of size 2^2 shown in Fig. 6.2(b) has four switching elements and eight pins. The switching elements of each subnetwork in the first two stages are marked with the same letters as shown in Fig. 6.2(a) and those of the subnetwork in the last two stages are shown in dash lines. The total pin number required in the partition of Fig. 6.2 is much less than that required in the partition of Fig. 6.1. To reduce the total pins required, we prefer the partition scheme whose composite subnetwork has switching elements for different stages. The partition shown in Fig. 6.2 can be expressed as $n=2+2$. Fig. 6.3 shows another partition example. As shown in Fig. 6.3(a), the first stage is implemented by eight subnetworks of size 2^1 and the last three stages are implemented by two subnetworks of size 2^3 which are shown in the blocks of dash lines. The composite subnetwork of size 2^3 and 2^1 are shown in Fig. 6.3(b) and (c), respectively. The partition is then $n=1+3$. The last partition example is shown in Fig. 6.4. The first two stages are implemented by four subnetworks of the type shown in Fig. 6.2(b) and the last three stages are implemented by two subnetworks of the type shown in Fig. 6.3(b). Again, the switching elements of each subnetwork in the first two stages are marked with the same letters and those of the subnetwork in the last three stages are shown in dash lines. The second stage of the network is repeated in the implementation. The

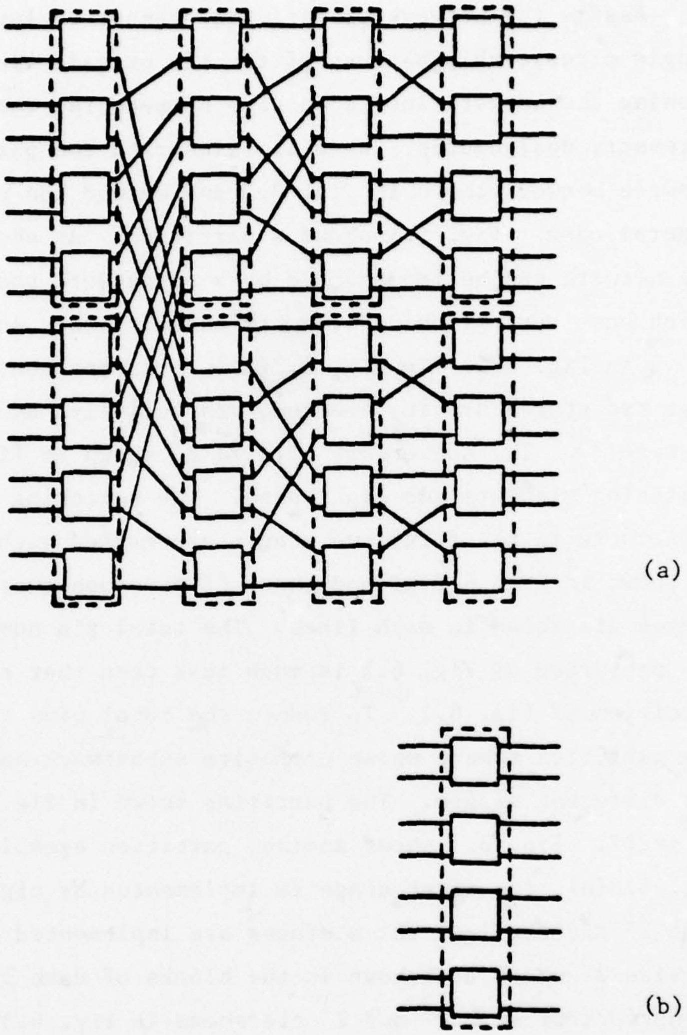


Fig. 6.1 The first partition example.

(a) The partition.

(b) Subnetwork for the partition.

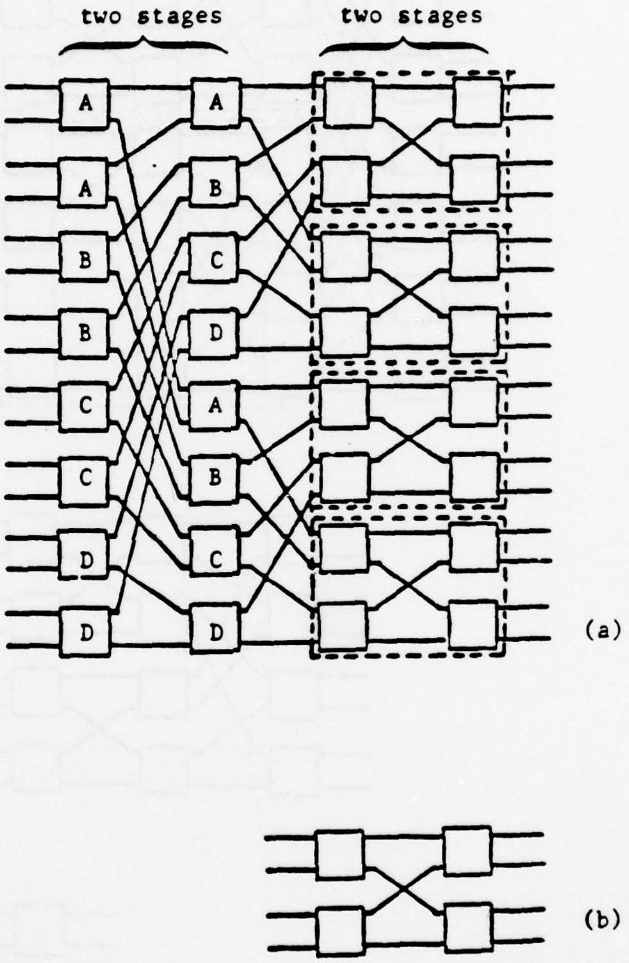


Fig. 6.2 The second partition example .

(a) The partition.

(b) Subnetwork for the partition .

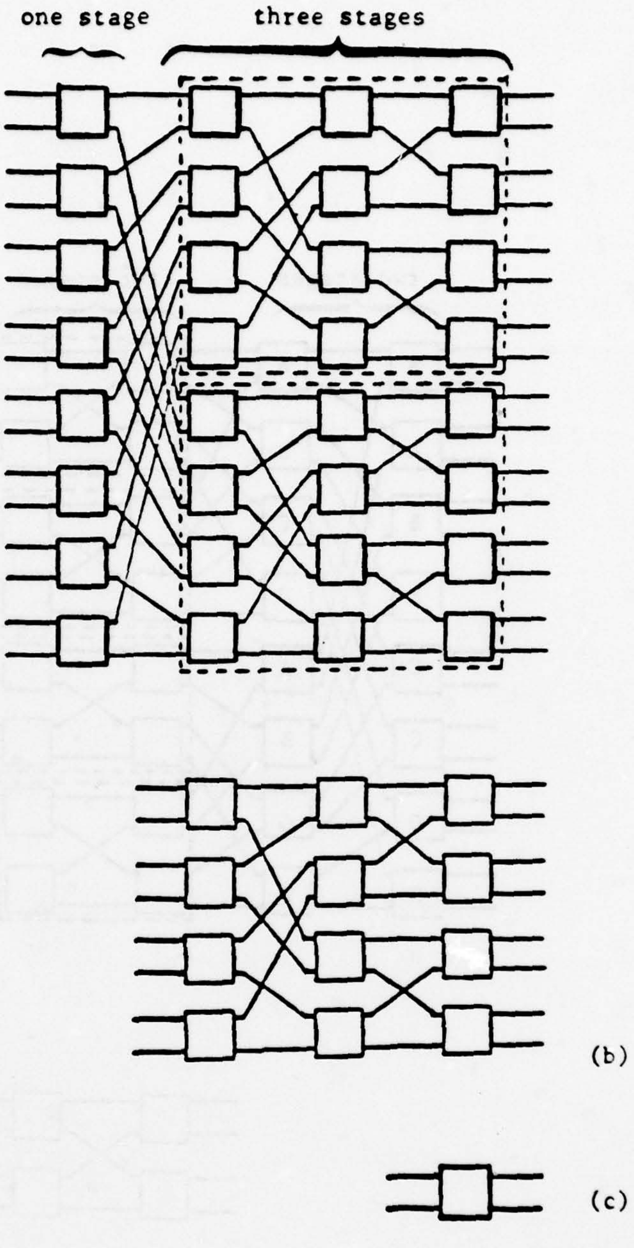


Fig. 6.3 The third partition example.

(a) The partition.

(b) Subnetwork for the last three stages.

(c) Subnetwork for the first stage.

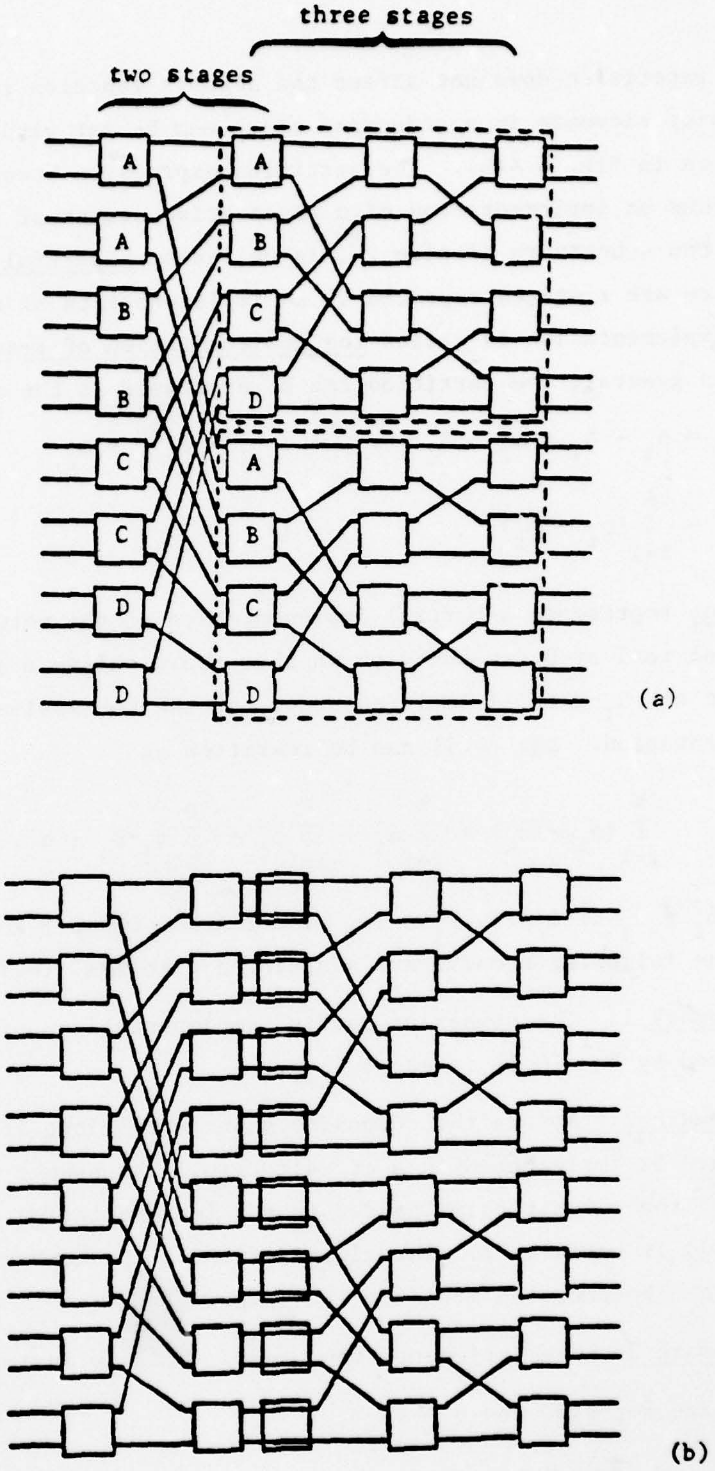


Fig. 6.4 The fourth partition example.

(a) The partition.

(b) Interconnection associated with the partition.

stage repetition does not affect the network function if the 2 × 2 switching elements in a redundant stage can be set without exchange as shown in Fig. 6.4(b). The partition expression becomes n=2-1+3.

Thus an implementation of α consecutive stages of the network, using the subnetwork of size 2^{α}, is called an <u>α-partial implementation</u>. If there are r stages repeated in an implementation of the network, this implementation is called <u>the implementation of r-stage repetition</u>.

In general, the partition can be expressed as the following:

$$\begin{aligned} n &= \alpha_1 - \delta_1 + \alpha_2 - \delta_2 + \ldots + \alpha_{k-1} - \delta_{k-1} + \alpha_k \\ &= \sum_{i=1}^{k} (\alpha_i - \delta_i) \quad , \end{aligned} \tag{6.1}$$

where α_i represents a partial implementation of the network and δ_i is equal to 1 or 0 depending on whether there exists a stage repetition between the α_i-partial implementation and the successive partial implementation. Eq. (6.1) can be rewritten as

$$\sum_{i=1}^{k} (\alpha_i - \delta_k) = \sum_{i=1}^{k} \alpha_i - \sum_{i=1}^{k} \delta_i = \sum_{j=1}^{p} m_j \cdot B_j - q \ , \tag{6.2}$$

where $\beta_i \neq \beta_j$ if $i \neq j$, $B_j \varepsilon \{\alpha_i \mid 1 \leq i \leq k\}$, $0 < p \leq k$ and $0 \leq q < k$.

The following remarks are associated with Eqs. (6.1) and (6.2).

<u>Remark 1</u>: The number of partial implementations of the partition expressed by Eq. (6.1) is equal to k.

<u>Remark 2</u>: Assume the composite subnetwork along its control structure is implemented in a circuit chip. The number of the modular types of the circuit chips needed in the implementation expressed by Eq. (6.1) is equal to p. The minimization of the number of the modular types then becomes the minimization of p.

<u>Remark 3</u>: The implementation of Eq. (6.1) is an implementation of q-stage repetion and $q < \sum_{j=1}^{p} m_j$.

<u>Remark 4</u>: The subnetwork needed in an α_i-partial implementation is a $2^{\alpha_i} \times 2^{\alpha_i}$ baseline network which can be implemented in a circuit

chip and the number of chips needed is equal to $2^{n-\alpha_i}$.

Remark 5: There are $\alpha_i 2^{\alpha_i - 1}$ 2 × 2 switching elements in the subnetwork needed in an α_i-partial implementation.

6.2 Minimizing the Number of Modular Types

An approach to implementing a network of any size N ($N=2^n$) with circuit chips of one single modular type is shown here.

Theorem 6.1: A multistage network of any size N ($N=2^n$) can be implemented by using circuit chips of at most two modular types.

Proof: For any integer $\alpha \leq n$, it is always possible to express n in terms of α as shown in the following way:

$$n = Q \cdot \alpha + R, \qquad (6.3)$$

where $0 \leq R < \alpha$, and $Q \cdot \alpha$ represents α network repeated Q times.

It is a trivial case for n=1, since the network can be implemented by using the circuit chip containing the subnetwork shown in Fig. 6.3(c). For $n \geq 2$, according to Eq. (6.3), we can always find an α and an R, where $2 \leq \alpha \leq n$ and $0 \leq R < \alpha$ such that the network can be implemented by Q α-partial implementations and one R-partial implementation. The α-partial implementations can be realized by the circuit chip containing a $2^\alpha \times 2^\alpha$ baseline network and the R-partial implementation can be realized by the circuit chip containing the $2^R \times 2^R$ baseline network. If R is equal to zero, then only one modular type is needed for the implementation. Q.E.D.

Due to the limitation on the numbers of pins and/or gates allowed in an LSI chip, the size of a circuit chip used in a partial implementation should be confined in the allowable range.

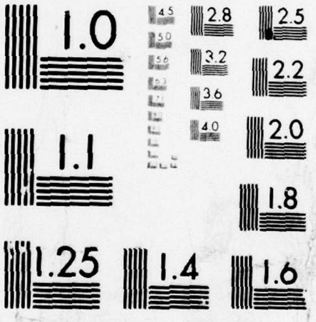
Assume the circuit chip containing a $2^{\alpha_m} \times 2^{\alpha_m}$ baseline network and the related control logic is the maximally allowable one. The α_m-partial implementation is called the maximum partial implementation.

The network whose size is less than 2^{α_m} can be implemented in a circuit chip. The implementation of the network whose size is larger than 2^{α_m}, with circuit chips of one single modular type, is described in the following theorem.

AD-A080 959 WAYNE STATE UNIV DETROIT MICH F/G 9/2
INTERCONNECTION NETWORKS IN MULTIPLE-PROCESSOR SYSTEMS.(U)
DEC 79 T FENG, C WU F30602-76-C-0282
UNCLASSIFIED RADC-TR-79-304 NL

3 OF 3
AD
A080959

END
DATE
FILMED
3-80
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Theorem 6.2: For the network whose size N ($N=2^n$) is larger than 2^{α_m}, i.e., $n > \alpha_m$, there exists a maximum α, $(\alpha_m)/2 < \alpha \leq \alpha_m$, such that the network can be implemented by α-partial implementations which can be realized by using circuit chips of one single modular type.

Proof: The theorem can be proven by showing that there exists a partition expression such as

$$n = t \cdot \beta - q, \tag{6.4}$$

where $(\alpha_m)/2 < \beta \leq \alpha_m$ and $q < t$. Eq. (6.4) implies that the network can be implemented by t α-partial implementations with q stage repetitions. This implementation can be realized by using circuit chips which are of the same modular type. The circuit chip contains a $2^\beta \times 2^\beta$ baseline network and the related control logic.

By Eq. (6.3) we have

$$n = Q_m \cdot \alpha_m + R_m , \tag{6.5}$$

where $0 \leq R_m < \alpha_m$ and $Q_m \geq 1$. Eq. (6.5) can be transformed into

$$n = (Q_m+1)(\alpha_m-j) - [\alpha_m-R_m-j(Q_m+1)]. \tag{6.6}$$

The remaining proof work is to show that there exists an integer j with the property of $0 \leq j < (\alpha_m)/2$ such that

$$0 \leq \alpha_m-R_m-j(Q_m+1) \leq Q_m . \tag{6.7}$$

For the case of $\alpha_m - R_m \leq Q_m$ we can set j=0 and obtain the following equation from Eq. (6.6) :

$$n = (Q_m+1)\alpha_m - (\alpha_m-R_m) . \tag{6.8}$$

If we set $t=Q_m+1$, $\beta=\alpha_m$ and $q=\alpha_m-R_m$ we can obtain Eq. (6.4) from Eq. (6.8). For the case of $\alpha_m-R_m > Q_m$ we should set $j \geq 1$ in order to have $0 \leq \alpha_m-R_m-j(Q_m+1) \leq Q_m$. However j should be less than $(\alpha_m)/2$. We will verify the statement using proof by contradiction. Assume $j \geq \lceil(\alpha_m)/2\rceil$. Then

$$\alpha_m-R_m-j(Q_m+1) \leq \alpha_m-R_m-(\alpha_m/2)(Q_m+1) . \tag{6.9}$$

From Eq. (6.9) we have

$$\alpha_m - R_m - j(Q_m+1) \leq [\alpha_m - R_m - (\alpha_m)/2] - [(\alpha_m)/2]\, Q_m \quad . \tag{6.10}$$

Since $\alpha_m - R_m - (\alpha_m)/2 < (\alpha_m)/2$ and $Q_m \geq 1$, we have, from Eq. (6.10),

$$\alpha_m - R_m - j(Q_m+1) < (\alpha_m)/2 - (\alpha_m)/2 \ . \tag{6.11}$$

Eq. (6.11) implies

$$\alpha_m - R_m - j(Q_m+1) < 0 \ . \tag{6.12}$$

Eq. (6.12) contradicts Eq. (6.7). Hence

$$0 \leq j < (\alpha_m)/2. \tag{6.13}$$

Using Eq. (6.6) and (6.13) and setting $t=Q_m+1$, $q=\alpha_m - R_m - j(Q_m+1)$ and $\beta=\alpha_m - j$, we can at least obtain an expression of Eq. (6.4). There may be several values of j which can lead Eq. (6.6) to Eq. (6.4) and the least of those values makes the implementation use the largest circuit chips of one single mocular type. Q.E.D.

Example: Assume that the network size N is equal to 2^{14} (n=14) and $\alpha_m=6$. According to Eq. (6.3) we have $n=2\cdot 6+2$ which means that the network can be implemented by two six-partial implementations and one four-partial implementation. From Eq. (6.6) we have $n=(2+1)\cdot 6-4$ if we set j=0. Since $q(=4) > Q_m(=2)$, we have to set $j > 0$ in Eq. (6.6). Setting j=1, we can obtain $n=(2+1)\cdot 5-1$. It is an implementation of one-stage repetition. The implementation employs three five-partial implementations which can be realized by using circuit chips of one single modular type. The circuit chip contains a $2^5 \times 2^5$ baseline network and the related control logic. The total number of circuit chips needed is equal to $3\cdot 2^9$.

6.3 Analysis on Pins

We will count the total pin number of the circuit chips required in an implementation. A circuit chip needs pins for input and output terminals of the network body, for the control of switching elements, for the chip selection and for the power supply and ground. In an α-partial implementation each circuit chip has 2^α pins for input terminals and 2^α pins for output terminals. Usually, three pins are required in a circuit chip for the power supply and ground. The number of pins for the chip selection depends on the total number of circuit chips incorporated for the overall network. And the number of pins for the control of switching elements in a circuit chip depends on the control structure required for the network functions. However, how to implement the control structure is still subjected to investigation.

Some relationships between the control structure and the manipulating functions has been shown in [19], [38] and [43] for synchronous operations. Accordingly the control structure has been classifiedinto three categories in [50]: individual stage control, individual box control and partial stage control. The realization of the control structure depends on the capability of the LSI technology, the response time requirement and the routing techniques, etc. Besides the synchronous operation, the asynchronous operations are also useful for the multiprocessing and distributed processing. Taking the control structure into account, we design two modules for our analysis on pins, one for the asynchronous operations and the other for the synchronous operation with the individual stage control.

A. Design of a Module for the Asynchronous Operation

First, the control structure of the 2 x 2 switching element for the asynchronous operation is subjected to investigation. Fig. 6.5 shows seven possible valid states for the 2 x 2 switching element. Three control bits must be used to implement these seven valid states. However, these seven states can be divided into two equivalent sets: {a,b,c,f} and {a,d,e,g}. The former set can be implemented by state f without conflicting the functional implementation of each valid state in the set and the latter by state g. An appropriate input and output mask scheme can be used to prevent routing errors induced by unused paths. Hence, instead of using three control bits for the seven valid states, we use only one control bit, c, to implement state f (c=0) and state g (c=1). Next, we will consider the routing problem in a module containing a network of size 2^{α}. Let $\ell=\alpha-1$. Assume that a source link $A = a_{\ell}a_{\ell-1}\cdots a_0$ on the left side of the network is to be connected to a destination link

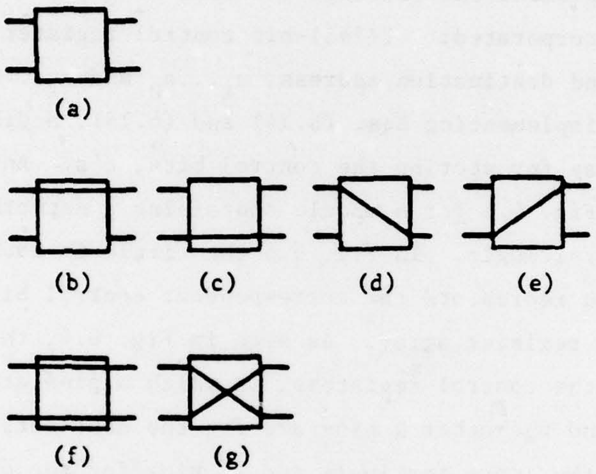


Fig. 6.5 Seven valid states in the asynchronous operation.

$Z = z_\ell z_{\ell-1} \ldots z_0$ on the right side of the network. From the previous result, the set of 2 × 2 switching elements which are in the connected path is

$$S = \{(z_\ell \ldots z_{\ell-i+1}\ a_\ell a_{\ell-1} \ldots a_{i+1})_i \mid 0 \leq i \leq \ell\}. \quad (6.14)$$

The control bit, c, of switching element $(z_\ell \ldots z_{\ell-i+1}\ a_\ell a_{\ell-1} \ldots a_{i+1})_i$ can be sent as

$$c = z_{\ell-i} \oplus a_i. \quad (6.15)$$

Hence, to implement the routing, the following hardware feature should be incorporated: $2(\ell+1)$-bit control register for storing the source and destination address, $a_\ell \ldots a_0$ and $z_\ell \ldots z_0$, a decoder for implementing Eqs. (6.14) and (6.15), a distribution register array for storing the control bits, c's. An example is shown in Fig. 6.6 for a module containing a network of size 2^4 and the control logic. In Fig. 6.6 the circle in each switching element block represents the correspondent control bit of the distribution register array. As seen in Fig. 6.6, there are 2^α pins for the control registers, in which α pins are for the source tag and the other α pins are for the destination tag, 2^α pins for the input terminals and 2^α pins for the output terminals. Besides, three pins are needed for power and ground, and a few for the chip selection which depends on the number of circuit chips incorporated for the overall network. Hence the total number of pins needed in the module is equal to $2^{\alpha+1} + 2\alpha + 3 + s$, where s is the number of pins needed for the chip selection.

B. Design of a Module for the Synchronous Operation with Individual Stage Control

Again we will consider the design of a module containing a network of size 2^α. The individual stage control uses the same control line for all switching elements in the same stage. Since there are α stages of switching elements, we can use α pins each of which feeds routing information into a control bit for a specified stage. Including the pins for the chip selection, for the power supply and ground, and for the terminals, we have

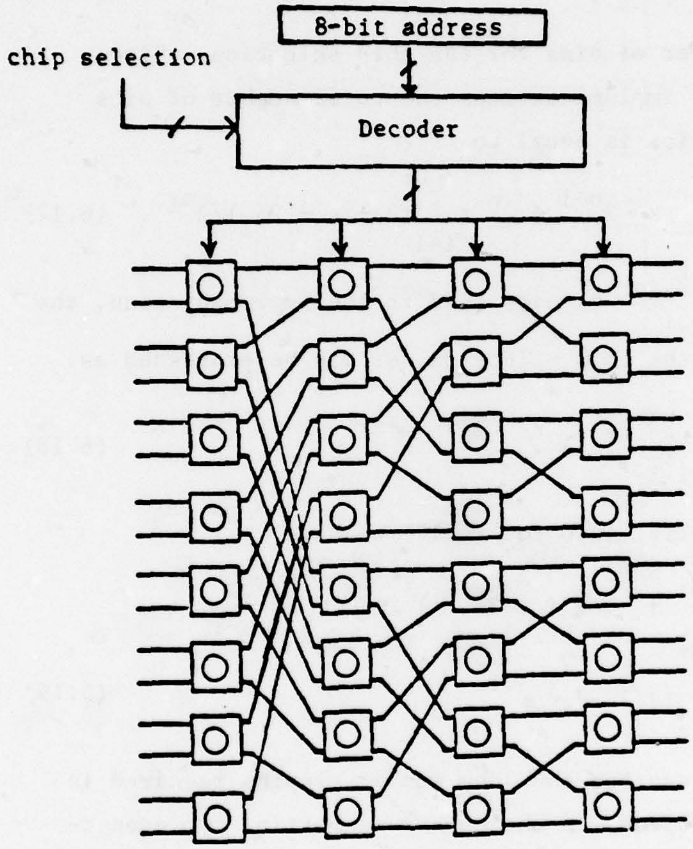


Fig. 6.6 **A module for the asynchronous operation.**

$2^{\alpha+1} + \alpha + 3 + s$ pins in the module, where s is the number of pins needed for the chip selection.

Consider the implementation for the asynchronous operation first. Since the total number of pins of a circuit chip which contains a $2^{\alpha} \times 2^{\alpha}$ baseline network is equal to $2^{\alpha+1} + 2\alpha + 3 + s$, we can express the total pin number of an α_i-partial implementation as

$$p_i = (2^{\alpha_i+1} + 2\alpha_i + 3 + s) \cdot 2^{n-\alpha_i}, \qquad (6.16)$$

where s is the number of pins for the chip selection. Since there are k partial implementations the total number of pins for the implementation is equal to

$$P = \sum_{i=1}^{k} p_i = k \cdot 2^{n+1} + 2^n \cdot \sum_{i=1}^{k} (3 + s + 2\alpha_i)/2^{\alpha_i} . \qquad (6.17)$$

Because there are $\sum_{i=1}^{k} 2^{n-\alpha_i}$ chips used in the implementation, the number of pins for the chip selection, s, can be expressed as:

$$s = \log_2 (\sum_{i=1}^{k} 2^{n-\alpha_i}) . \qquad (6.18)$$

Substituting Eq. (6.18) into Eq. (6.17) we have

$$P = k \cdot 2^{n+1} + \log_2 (\sum_{i=1}^{k} 2^{n-\alpha_i}) \cdot 2^n \cdot \sum_{i=1}^{k} \frac{1}{2^{\alpha_i}} + 2^n \cdot \sum_{i=1}^{k} (3 + 2\alpha_i)/2^{\alpha_i}. \qquad (6.19)$$

From Eq. (6.19) we can see that the number of pins required in an implementation depends on the number of partial implementations, k, the size of the partial implementation, α_i, and the network size, 2^n.

Similarly, we can obtain the number of pins required in the implementation for the synchronous operation with the individual stage control.

Since there are $n\ 2^{n-1}$ 2 × 2 switching elements required in the definition network, the ratio of P and $n\ 2^{n-1}$, called , can be used as a measurement on the cost effectiveness instead of using the gate-to-pin ratio. Specifically,

$$\sigma = P/(n \cdot 2^{n-1})$$

$$= \frac{2}{n}\,[2k + \log_2\left(\sum_{i=1}^{k} 2^{n-\alpha_i}\right) \cdot \sum_{i=1}^{k} \frac{1}{2^{\alpha_i}} + \sum_{i=1}^{k} (3 + 2\alpha_i)/2^{\alpha_i}] \; . \tag{6.20}$$

In the sense that the less of the value σ the more the cost effectiveness of the implementation, we can obtain an optimal LSI implementation by comparing the σ values of candidate implementation.

6.4 Interconnecting Circuit Chips

The chip interconnection problem can be defined as that of properly interconnecting the output terminals of a partial implementation to the input terminals of the successive partial implementation such that the implementation results in a network whose overall interconnection pattern between two adjacent stages (excluding those between the stage and its repeated stages) can be described by the topology describing rules of the definition network.

The chip interconnection problem of two consecutive partial implementations can be considered in two cases depending on whether there is a stage repetition between these two partial implementations or not. In the case of having a stage repetition, the output terminals of the first partial implementation, named $(p_\ell p_{\ell-1} \cdots p_0)_i$, is connected to the input terminals of the successive partial implementation, also named $(p_\ell p_{\ell-1} \cdots p_0)_i$. In the case of having no stage repetition, the topology describing rules shown in Eqs. (3.1) and (3.2) can be used to connect the two partial implementations. However, in both cases, the binary representation of switching elements in the two partial implementations must be identified before the interconnecting can proceed. Hence the chip interconnection problem becomes the problem of the binary name assignment on the switching elements in a circuit chip.

The problem of the binary name assignment on the switching element in a circuit chip will be solved here. Assume stages i, i+1, ..., $i+\alpha-1$ in the network of size $N=2^n$ are implemented by an

α-partial implementation consisting of $2^{n-\alpha}$ circuit chips. Each of these chips contains a $2^\alpha \times 2^\alpha$ baseline network. These circuit chips can be aligned side by side and the switching elements in the same stage line up in a column. The circuit chips are named by the sequence from 0 to $2^{n-\alpha}-1$ with 0 for the circuit chip on the top and $2^{n-\alpha}-1$ for the circuit chip in the bottom. There are two kinds of names which can be associated with a switching element. One is the physical name which identified the location of a switching element in the circuit chip. Another is the logical name which identifies the logic position of a switching element in the definition network and is used in the topology describing rules. The physical names of the switching elements in a circuit chip can be obtained by labelling the switching elements according to the position order in the same stage.

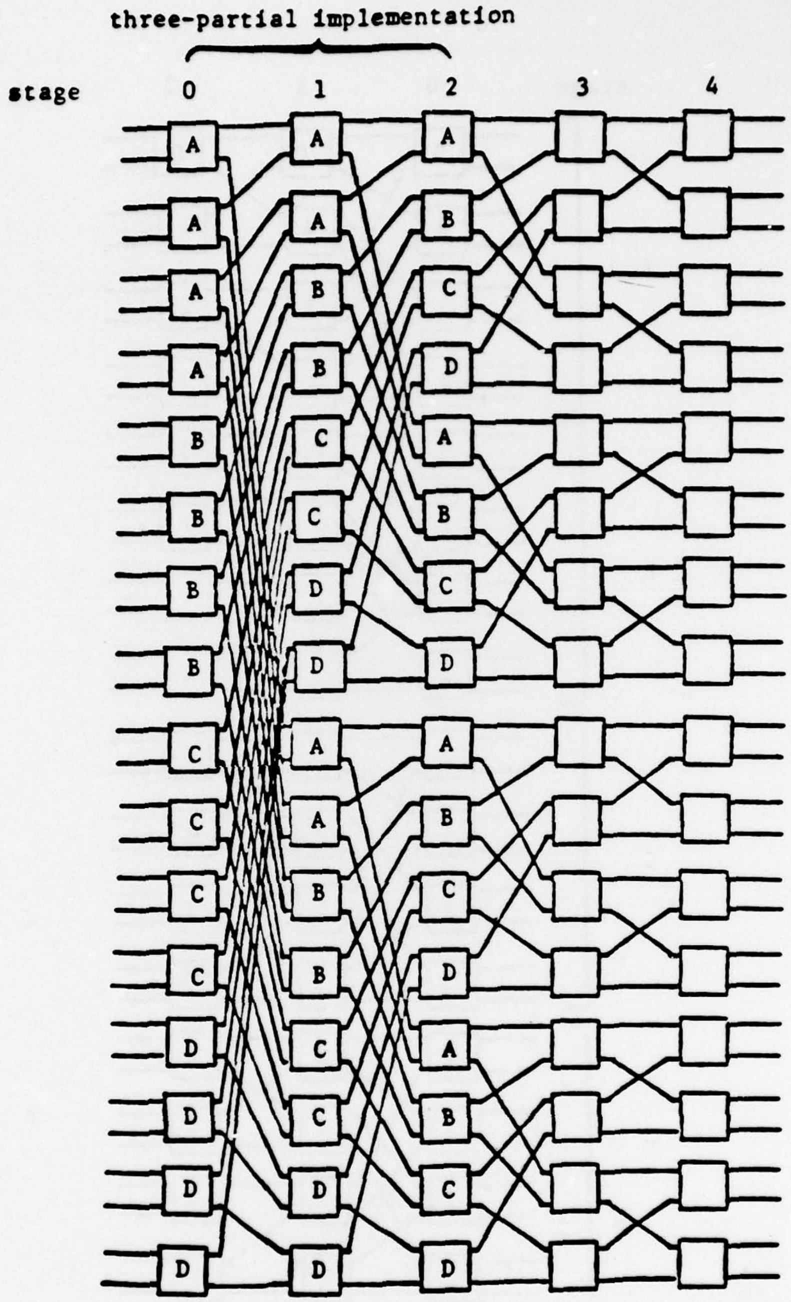
Example: For circuit chips of the same type which implement stage 0, 1 and 2 of the network shown in Fig. 6.7(a) are aligned side by side as shown in Fig. 6.7(b). The switching elements in the same chip are identified by the same letter in Fig. 6.7(a). The physical names of the switching elements in each circuit chip are also shown in Fig. 6.7(b) by decimal numbers.

Our problem is then to find the logical name, $(b_\ell b_{\ell-1}\cdots b_1)_{i+j}$, of the switching element whose physical name is $(p_{\ell-1}p_{\ell-2}\cdots p_1)_j$ in circuit chip k, where $0 \le j < \alpha$, $0 \le k < 2^{n-\alpha}$ and $n=\alpha+1$. Assume that $a_{n-\alpha}\cdots a_1$ is the binary code word of k. The logical name of that switching element can be expressed as:

$$(b_\ell b_{\ell-1}\cdots b_1)_{i+j} = \qquad (6.21)$$

$$(a_{n-\alpha}\cdots a_{n-\alpha-i+1}\; p_{\alpha-1}\cdots p_{\alpha-j}\; a_{n-\alpha-i}\cdots a_1 p_{\alpha-j-1}\cdots p_1)_{i+j}\,.$$

Example: The logical name assignment on the switching elements is shown in Fig. 6.8 for the partial implementation of the network shown in Fig. 6.7. In the partial implementation, n=5, α=3, i=0, $0 \le j \le 2$, and $0 \le k \le 3$. Plugging these numbers into Eq. (6.21) we obtain the logical names as shown in Fig. 6.8(b) which are one-to-one correspondent to those shown in Fig. 6.8(a).


three-partial implementation
stage
0
1
2
3
4
A
A
A
A
B
B
B
B
C
C
C
C
D
D
D
D
A
A
B
B
C
C
D
D
A
A
B
B
C
C
D
D
A
B
C
D
A
B
C
D
A
B
C
D
A
B
C
D

6.7(a)

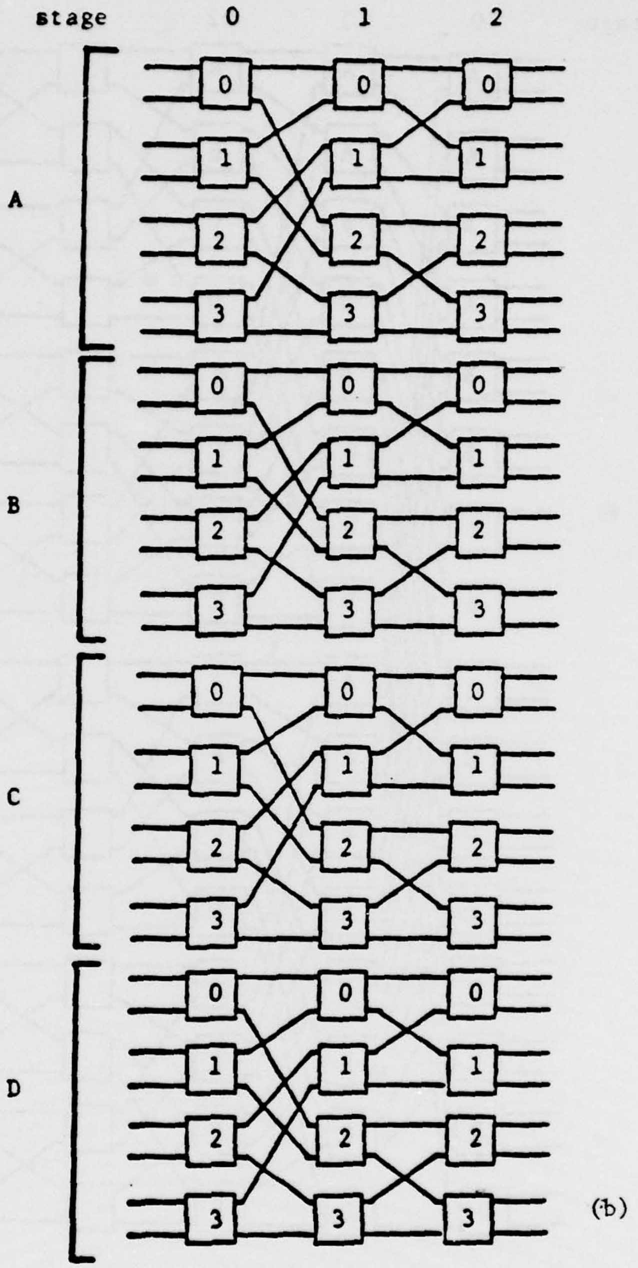
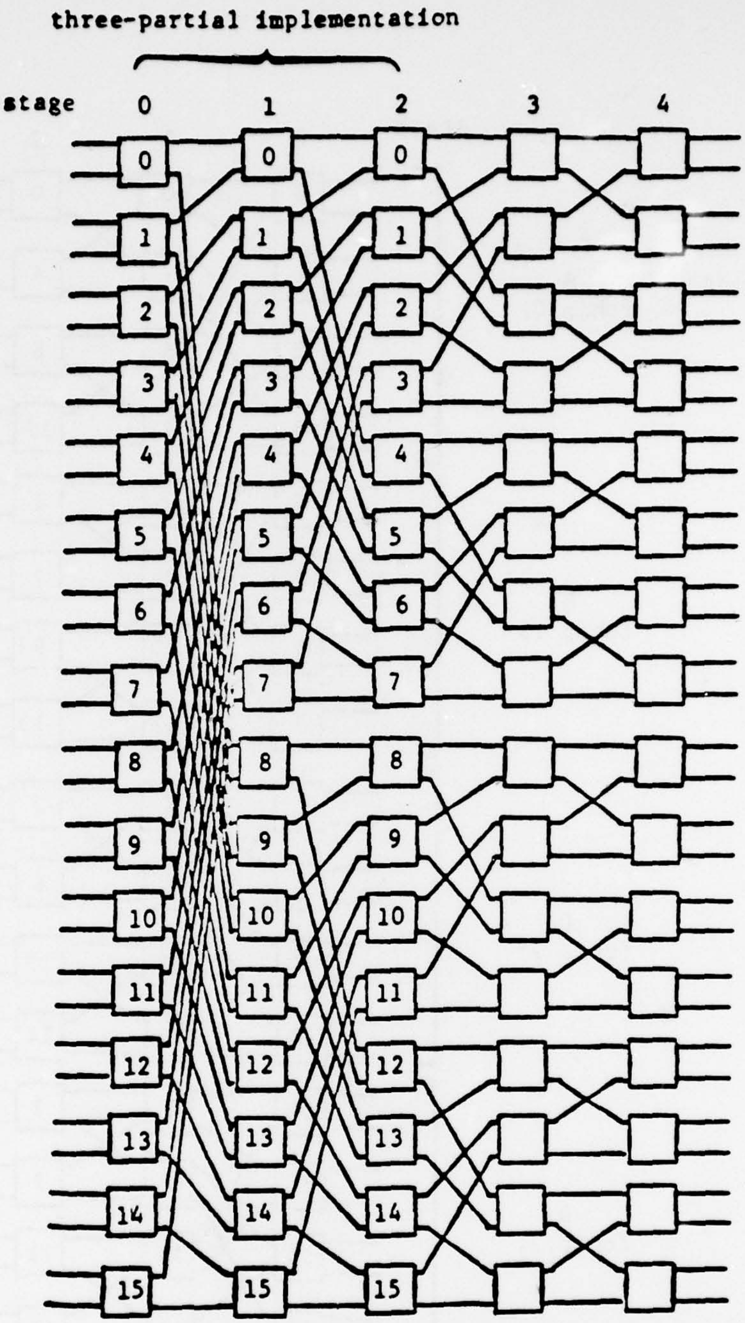


Fig. 6.7 Example for the partial implementation.

(a) A three-partial implementation.

(b) Subnetworks aligned side by side.


three-partial implementation
stage
0
1
2
3
4
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

6.8(a)

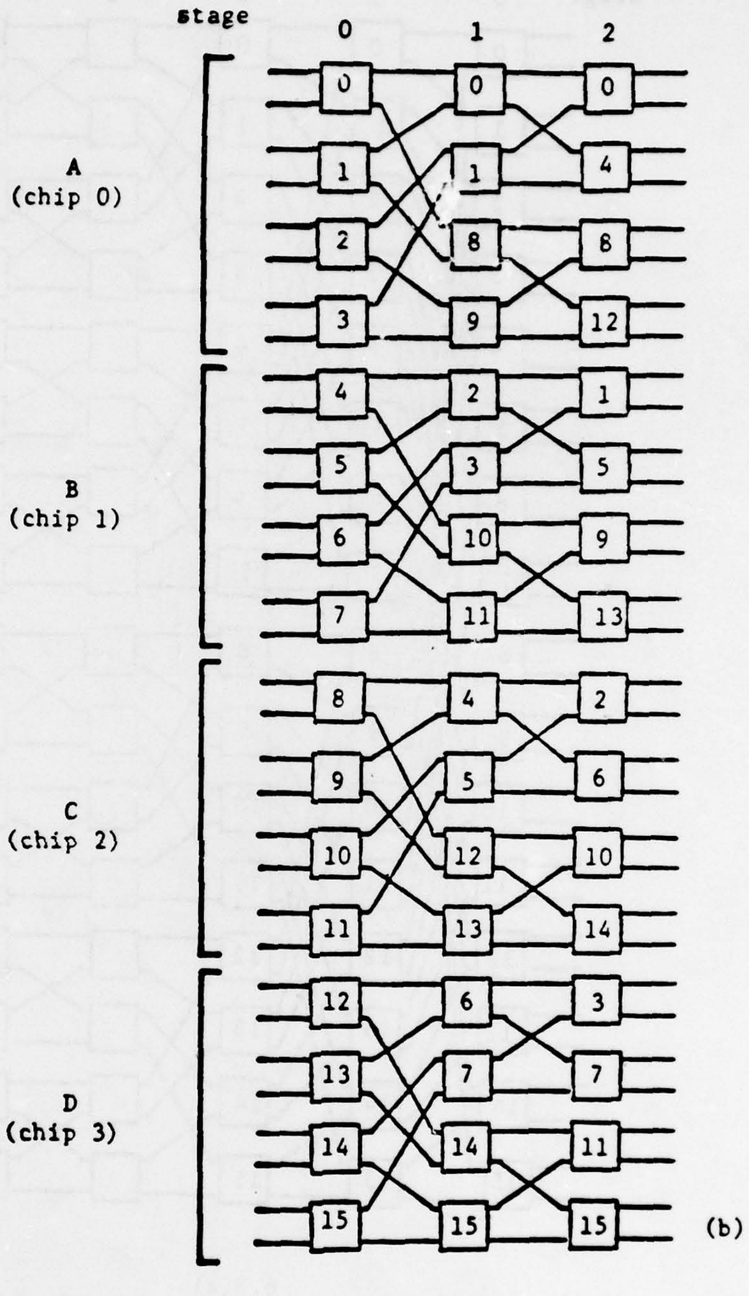


Fig. 6.8 Name assignment in the partial implementation.
(a) Logical names in the definition network.
(b) Logical names in the subnetworks.

6.5 Summary

In this chapter we have shown a logic partitioning scheme which can be used to implement a class of multistage interconnection networks optimally in the sense of using LSI circuit chips of one modular type and resulting in the maximum switching element-to-pin ratio. The scheme can be divided into four major parts. The first part shows how to partition the network and results in a general partition formula. In the second part the general partition formula is manipulated to minimize the number of modular types. As shown in Theorem 6.2, if the maximum size of the network which can be implemented in an LSI circuit chip is equal to 2^{α_m}, we can always implement the baseline network of size 2^n, $n \geq \alpha_m$, using circuit chips of one type, each of which contains a baseline network of size 2^α where $(\alpha_m)/2 < \alpha \leq \alpha_m$. In the third part we count the total pins needed in an implementation partitioning. The last part tackles the problem of interconnecting circuit chips to fulfill the topology describing rules which define the network structure. A formula has been developed to identify switching elements in each circuit chip.

CHAPTER 7
CONCLUSION

The problem of interconnecting units in a multiple-processor system is receiving increasing attention. The interconnection organizations of time-shared/common buses, crossbar switches and multiport memory schemes have their limitations when the number of functional units in the system becomes large because of the impact of recent advances in LSI technology and the projected processing requirements. In this respect, the multistage interconnection networks have been considered as good candidates for the interconnection organization. However, previous works on the multistage interconnection networks generally relate only to the network topology or the implementable permutation functions. A few investigators have considered routing algorithms. However, the important network such as Benes binary network has been ruled out because the routing algorithms for the Benes network are not fast enough for highly parallel computer. The future technology evolution has not been taken into account. Furthermore, most of the authors have claimed that their network is the best without providing strong evidences. Hence, reviewing previous works, we feel that this important field is still lacking a set of performance standard and evaluation tools which can be used to observe the tradeoffs among various parameters. In addition, few investigators consider the communication and the design of the interconnection network as a whole and the influence of communication protocols which should, nevertheless, be implemented for the intercommunication function of the interconnection network is usually neglected. Besides these problems, the fault diagnosis scheme for the interconnection networks, which is important for a reliable or fault tolerant system, has not been developed and the multiple-pass realization of an interconnection network and related routing algorithms have only been discussed for a single-stage network. The problem concerning the LSI implementation of the interconnection networks also remains unanswered.

Chapter 2 surveys the multiple-processor intercommunications. Firstly, we review the interconnection organizations of multiple-processor systems to emphasize the importance of multistage interconnection networks by showing the limitations of interconnection organizations such as time-shared/common buses, crossbar switches and multiport memory schemes. Then we survey particular multistage interconnection networks which were proposed from significantly different viewpoints. Generally speaking, the nonblocking characteristic of the interconnection networks is not the prime criterion for choosing network structure. Instead, blocking networks with uniform structure are always found in practical usage and proposed schemes. Among these blocking networks, the complexity, the number of switching elements and stages do not differ significantly between approaches. An important step in designing an interconnection network is to choose a network structure which can facilitate implementing effective communication protocols. In the latter part of Chapter 2, we catalogue a wide variety of switching concepts and parameters which a designer may have to encounter in planning and designing an interconnection network. In the last part of Chapter 2, we provide a set of characteristics of interconnection networks, which can be used to specify the performance standard, and describe some hardware and software requirements for implementing functions of interconnection networks.

A baseline network is introduced in Chapter 3 to evaluate the relationships among the multistage interconnection networks which have been proposed from significantly different viewpoints. It is shown that a class of topologically equivalent multistages interconnection networks can be obtained by properly permuting the switching elements and associated links of the baseline network within the same stage. The class of topologically equivalent multistages interconnection networks includes the indirect binary n-cube network, the modified data manipulator, the flip network, the omega network, the regular SW banyan network with S=F=2, the reverse baseline network, and the baseline network. A logical name representation scheme is developed to configure this class of the topologically equivalent networks. It is shown that one network in this class can share the same routing information developed for another network in this same class if these two networks use the same representation scheme.

The logical name representation scheme enables a simple routing algorithm and the routing algorithms are proven to be complete and homogeneous so that no distinction should be made between the inputs and the outputs. A routing procedure is developed on the basis of the homogeneous routing algorithm. Since all the networks in the defined class are blocking, the routing procedure includes the capability to resolve the conflicts by choosing a deferred set of mapping requests according to some priority scheme. The routing procedure can also be extended to allow any connections between all pairs of terminals so that there is no need to divide the terminals into two disjoint sets.

The logic name representation scheme provides a formal addressing for the components of the interconnection networks. Using this addressing sheme, we can then identify the design issues for the fault-diagnosis scheme, the logic partitioning, and the packet switching communication. The homogeneous routing procedure and the full communication already provide a routing protocol for the packet switching communication.

In Chapter 4 we present a fault model for the network in the class of multistage interconnection networks. Fault diagnosis procedures for the network constructed of switching elements with two valid states have been considered. A diagnosis method for single faults and a detection method for multiple faults are developed. In the diagnosis procedures the control lines of the switching elements in the same stage can be grouped together and activated by the same control signal. The control line grouping of each stage is exactly the control scheme used in the flip network of STARAN. Hence, the diagnosis procedures developed in this paper are good both for the indirect binary n-cube network and the flip network. Extension to the network constructed of switching elements with four valid states is feasible since the test sets of faults in switching elements with four valid states are the same as those we developed for switching elements with two valid states. The problem left is to design diagnosis procedures with minimal or nearly minimal number of tests.

The number of tests which is required under various conditions in the diagnosis procedures developed in Chapter 4 is summarized as follows. The number of tests for detecting single faults is equal to four and is independent of the network size. The number of tests for detecting multiple faults is equal to $2(1+\log_2 N)$, where N is the number of terminal links in one side of the network. The number of tests needed for determining the fault lccation and the fault type of a single fault depends on the fault type and/or the size of the network. The minimum number of tests needed for determining the fault location and the fault type is equal to four and the maximum $\max(12, 6+2\lceil \log(\log N)\rceil)$. For a network of size $N = 1024$ the maximum is equal to 14. There exist four switching element faults (Subcase F) which cannot be pinpointed at the single switching element level and those four are not distinguishable from the link stuck fault. This study provides specific information of fault characteristics for designing an easily diagnosable network, or a fault tolerant network.

A reverse-exchange interconnection network that is shown to be a powerful interconnection network for the parallel processing system is introduced in Chapter 5. The homogeneous routing procedure can be used to control the network for general permutations. We have also provided a set of theorems to specify the realizable permutations which are useful to the parallel processing. The realizable permutations are then classified into four groups and a recursive formula is derived to calculate the control pattern of the network for each permutation group. The recursive formulas can provide superior operating speed over the existing routing algorithms.

It is proven that all permutations can be realized by the reverse-exchange network in two passes. Both the construction and the routing algorithm are provided. Our result compares favorably with $O(\sqrt{N})$ or $O(\sqrt{N}\log_2 N)$ steps needed in other networks. The network is also shown to be useful for the bit-reverse permutation, the multi-dimensional access memory and partitioning the array computer.

The needs of having a cost effective LSI implementation of interconnection networks are obvious. Among the needs are to extend the switch design from size 2 × 2 to size $2^{\alpha} \times 2^{\alpha}$, to improve the reliability of switching elements, and to expose problems in LSI implementation of interconnection networks. However, there is little activity concerning these needs. In Chapter 6, we have shown a logic partitioning scheme which can be used to implement the class of multistage interconnection networks optimally in the sense of using LSI circuit ships of one modular type and resulting in the minimum pin-to-switching element ratio. The scheme can be divided into four major parts. The first part shows how to partition the network and results in a general partition formula. In the second part the general partition formula is manipulated to minimize the number of modular types. As shown in Chapter 6, if the maximum size of the network which can be implemented in an LSI circuit chip is equal to 2^{α_m}, we can always implement the baseline network of size 2^n, $n \geq \alpha_m$, using circuit chips of one type, each of which contains a baseline network of size 2^{α} where $(\alpha_m)/2 < \alpha \leq \alpha_m$. In the third part we count the total pins needed in an implementation partitioning. The last part tackles the problem of interconnecting circuit chips to fulfill the topology describing rules which define the network structure. A formula has been developed to identify switching elements in each circuit chip.

In summary, we first describe general design philosophy of the interconnection networks and then define a class of multistage interconnection networks. For the class of multistage interconnection networks, we develop the routing algorithms, the fault-diagnosis scheme, the reverse-exchange permutation groups, and the two-pass realization of all permutations,and the logic partitioning for LSI implementation.

Several possible extensions which are closely related to this study appear to be:

(1) Implement packet switching in the class of multistage interconnection networks by using the homogeneous routing procedure.

(2) Extend the binary tree coding method to general interconnection networks.

(3) Use associative memory techniques to implement the routing procedure and the conflict resolution scheme.

(4) Design an easily testable switching element using the fault characteristics developed.

(5) Re-evaluate the fault-diagnosis scheme to locate fault at circuit module level.

(6) Extend the fault-diagnosis scheme to the networks with broadcasting capability such as the omega network, and to general networks such as Benes binary networks and crossbar networks.

(7) Investigate the possibility of using the reverse-exchange interconnection network for the dynamic memory access and block data access.

(8) Quantitatively compare the reverse-exchange network to the other networks such as the shuffle-exchange network and the flip network.

(9) Systematically partition the network control structure along with the network body for LSI implementation.

APPENDIX

A MICROPROCESSOR-CONTROLLED ASYNCHRONOUS CIRCUIT SWITCHING NETWORK

Abstract

This appendix describes an asynchronous circuit switching network for multiple-processor systems. Several circuit switching networks for various applications have been proposed and constructed. However, typical problems associated with these networks include that of the synchronous switching, the central control, the graceful degradation, the flood routing, the limited connectivity, and the cost effective LSI implementation and software development. The asynchronous circuit switching network possesses several features that can solve such problems. A three-stage fully connected topology is utilized to construct the network. Each switching element of the selected topology is functionally and physically identical and this facilitates a cost effective LSI implementation and software development. The control structure of the switching element and the routing algorithm are re-organized to fit the asynchronous operation. The network takes advantage of low-cost microcomputers to do the distributed routing control and to implement the communication protocols. The graceful degradation characteristic in the network is provided by independent multiple paths existing between any pair of the source and the destination. A network monitor is incorporated to facilitate an adaptive routing strategy and to have the fault disgnostic capability. Three alternatives for the switching element implementation are described to demonstrate the hardware and software trade-offs. The network architecture seems to facilitate a substantially high throughput intercommunication system for the tightly coupled distributed processing. The response time characteristic under various conditions is still to be verified by simulation studies.

A.1 Introduction

The interconnection organization of processors is a key to the classification of computer systems. Various attributes of interconnection organizations such as the transfer strategy, the control method, the path structure, and the system topology are used to classify the actual system designs [14]. The multiprocessor systems are also classified into three categories of interconnection organizations: time-shared/common-bus systems, crossbar switch systems and the multiport memory systems [13]. Basically these characterizations are derived from the existing systems. However recent advances in LSI technology have caused a significant change in the field of computer architecture. One trend is to use a plurality of homogeneous or heterogeneous processors interconnected together to gain operating power through parallelism and improve system reliability through redundancy. Based on this trend, various multiple-processor systems such as associative, parallel, pipeline, and multiprocessors are proposed and constructed [1]. The number of homogeneous or heterogeneous processors in the multiple-processor system will keep increasing due to several reasons. First, the processing speed in the future can be significantly increased only by increasing the degree of the concurrent processing. Furthermore, the low cost of LSI modules allows the use of a large number of processing elements. Thirdly, there are certain classes of problems, such as large data base management, weather computations, etc., which are beyond the capabilities of the current large computers. However when the number of processors in the multiple-processor system increases to a certain level, say the order of 100, the choice of interconnection organizations becomes a critical problem. People are even considering implementing a multiple-processor system by interconnecting as many as 10^5 processing units. System performance and practical feasibility of such a multiple-processor system would be terribly limited if the conventional interconnection technique is used. Thus one of the exciting challenges in the field of computer architecture is to design an efficient and practical intercommunication subsystem for multiple-processor systems.

Several new interconnection organizations have appeared in the literature. These include the flip network in the STARAN [19], the indirect binary n-cube network for a microprocessor array [45], a three-stage interconnection network for a communication processor proposed by North Electric Company [76], and a distributed data network for distributed processings [77]. Even in these new interconnection organizations there exist some crucial problems. In the flip network, the simple control structure allows only a few synchronous permutations. This shortcoming is overcome by the indirect binary n-cube network using individual control structure for each switching element. However, the central control strategy of the indirect binary n-cube becomes practically unfeasible when the number of processors is very large. Also, the flip network and the indirect binary n-cube network do not have the graceful degradation property as only one path exists between the source and the destination processors. The interconnection organization proposed by North Electric Company can provide multiple paths between source and destination. However, its flood routing scheme severely limits the availability of the components in the network and hence may cause significant response delay. The distributed data network using 24 links between sources and destinations allows simultaneous communication between 24 pairs of CPU's while the other 207 CPU's could possibly wait for their turn to transmit data. A full-connection network could better solve this bottleneck.

Beyond these interconnection problems of synchronous switching, central control, graceful degradation, routing techniques, and full connection, we have to consider the cost effective LSI implementation of the interconnection organization. For a cost effective LSI implementation, the minimization of the number of modular types is of prime importance and not the number of components utilized. Hence, it would be a good criterion to partition the interconnection organization into functionally and physically equivalent modules so that LSI implementation and software control programs developed for a single module can be used in all equivalent modules. On the other hand, according to the past progress in LSI technology, we may project that the most complex computer system of today can be fabricated on a small number of chips within the next few years [12].

Being moderate we can predict that it is feasible to fabricate a processor-memory-switch (PMS) group on a single chip which has several communication ports for the intercommunication purpose.

The objective of this study is to investigate the interconnection problems by using some multistage interconnection networks upon which the multiple-processor system can be modelled as shown in Fig. A.1. Here, the PMS box could represent any combination of processor, memory and switch, and IP is the interface processor. Section A.2 describes the configuration of an asynchronous circuit switching interconnection network which will be used to demonstrate the solutions to the interconnection problems. Section A.3 illustrates the general hardware structure. In Section A.4, we demonstrate the software control for the intercommunication with much emphasis given on the routing techniques. Finally Section A.5 discussed some system characteristics.

A.2 Network Configuration

The topology and the label of the component (stage, element, link) of the selected network are shown in Fig. A.2. The topology can be described by the definition of (N, N, N) Clos rearrangeable network [26], or series-parallel network [31]. The choice of the three-stage network is obvious. The full accessable single stage crossbar network has the disadvantages that there is only one path between a source-destination pair and the cost of the circuitry required for the switching facilities becomes significantly high when the number of network ports becomes large. There is only one path between a source-destination pair in the two-stage network and hence the graceful degradation is poor.

The stages are named by two-bit binary code words, 00, 01, and 10 from left to right. The left stage is connected to the active side and the right stage is connected to the passive side. The connection request can only be initiated on the active side. Assume N is a power of 2. The $N \times N$ switching element in each stage is named by $\ell = \log_2 N$ binary bits $p_\ell p_{\ell-1} \cdots p_1$ which are the binary representation of its location in the stage. Each interstage link is named by 2ℓ binary-bit code words $p_{2\ell} p_{2\ell-1} \cdots p_1$, which is coded

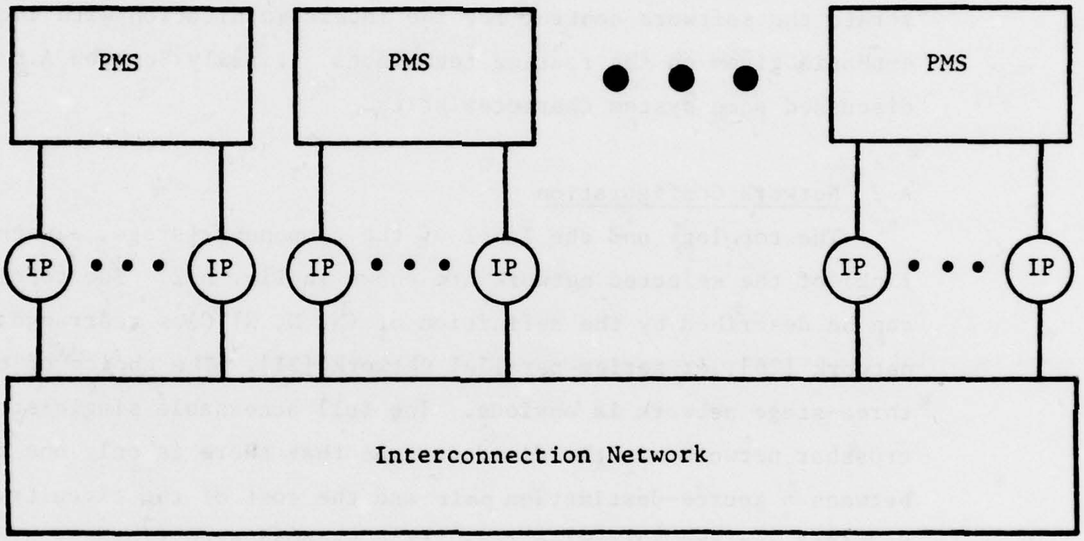


Fig. A.1 A model of a multiple-processor system. (IP = Interface Processor)

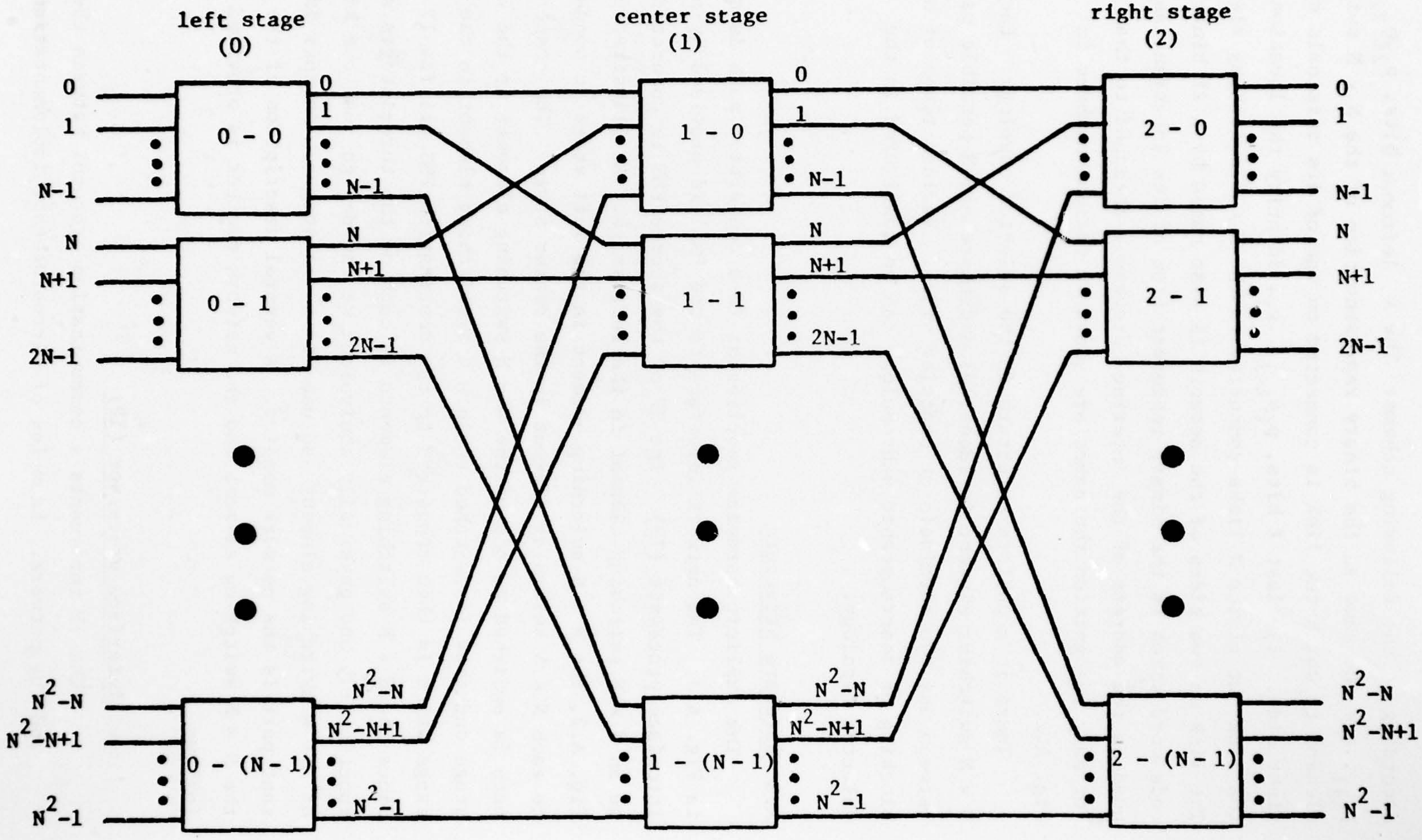


Fig. A.2 The network configuration.

according to the following scheme: The ℓ leftmost bits, $p_{2\ell}p_{2\ell-1}\cdots p_{\ell+1}$, are the same as the binary representation of the N × N switching element to which the link is connected on one of its terminals on the right side. The last ℓ bits, $p_{\ell}p_{\ell-1}\cdots p_1$, identify the location of the link out of the N links connected to the N × N switching element. The link on two sides of the network is also named by a 2ℓ binary-bit code word which is the binary representation of its location, and is used as the address of the interface processor attached to that link. For easy recognition the names are shown in decimal numbers in Fig. A.2.

There is a uniform structure in the selected topology. Each N × N switching element is identical and there are N possible paths between any two terminals at opposite sides. Various types of non-blocking or rearrangeable subtopology can be configured in the selected topology.

A.3 Hardware Structure

The explicit hardware requirement in a connection path is shown in Fig. A.3. The network interface to the PMS' is provided by an interface processor (IP). The IP of the source PMS is connected to an N × N switching element in the left stage. As illustrated in Fig. A.2, the N × N switching element in the left stage is connected to each N × N switching element in the center stage. The connection path is switched to one of the N × N switching elements in the center stage and then is switched to an N × N switching element in the right stage which is then connected to the destination PMS via its IP. Since the N × N switching elements in each of the three stages are functionally and physically equivalent we can design just one kind of N × N switching element and use it in all stages. Another important part is the network monitor. A general description of the IP, the N × N switching element and the network monitor is provided as follows.

A. Interface processor (IP)

The IP implements a communication protocol between the PMS and the network. Examples of microprocessor implementation of

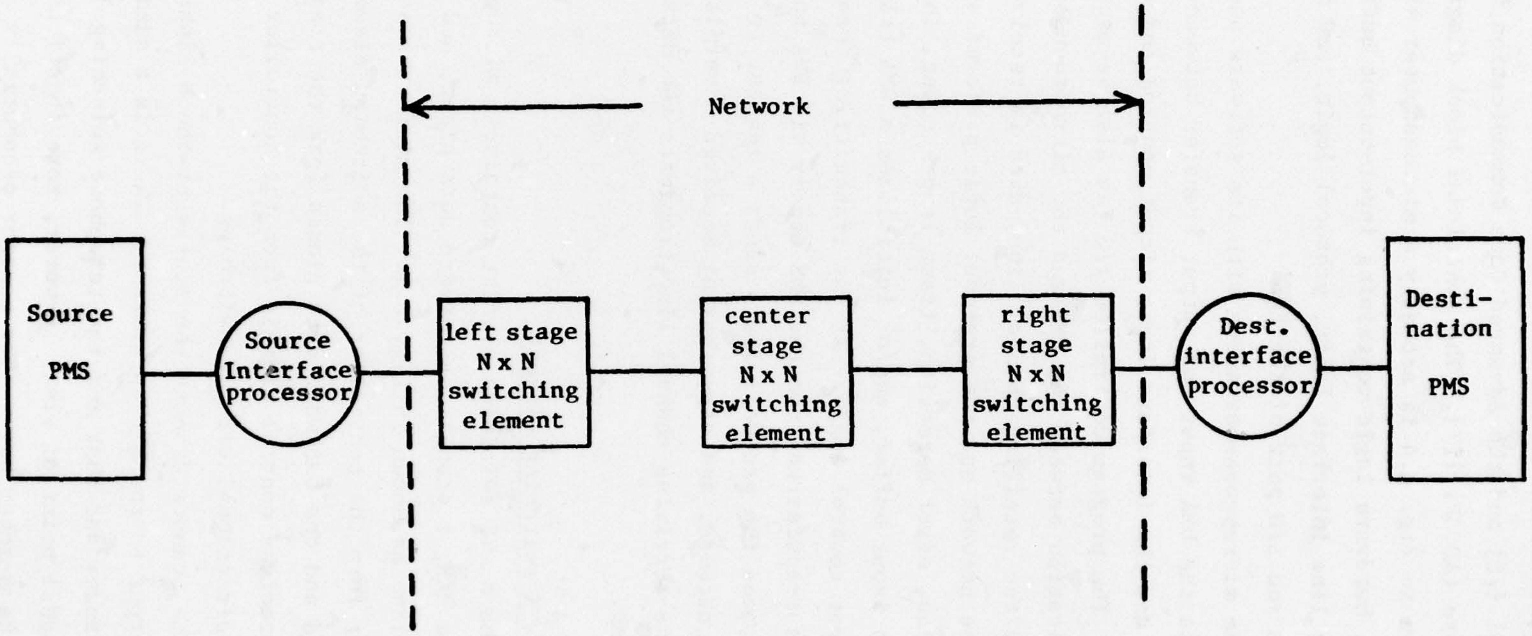


Fig. A.3 Explicit hardware requirement in a connection path.

the communication protocol are the CCITT recommended X.25 protocol [78] and the Advanced Data Communication Control Procedure (ADCCP) [77]. The functional block diagram of the IP shown in Fig. A.4 is actually a microcomputer with a set of special hardware logic containing input-output buffering logic, channel line interface logic, protocol logic, and interface for the DMA and I/O port of the PMS.

The microprocessor along with its software and firmware controls the DMA input and output transfer between the PMS and the IP and the interface logic (input control and output control). The program controlled I/O can also be used for the communication between the PMS and the microprocessor.

At the receiving channel line, data is received serially from the network and the protocol logic performs sequence detection, error detection, frame length count, loads the data into an input buffer, and/or initializes a DMA transfer via the input control logic. At the transmitting channel line, data is transferred from the PMS memory via DMA to an output buffer, and the protocol logic sends a request to gain access to the network, and after having received a permission message from the receiving channel line transmits the data serially over the link.

B. N × N switching element

The N × N switching element contains a microprocessor with ROM and RAM, a scanner, a forward data plane, and a backward data plane as shown in Fig. A.5. The microprocessor and the scanner form the controller of the switching element. The forward and the backward data planes form the transfer unit that receives control signals from the controller and provides the bidirectional circuit switching.

The scanner detects the status of the N links and sends the output to the microprocessor. There is a similarity between the scanner and that for the telephone switching [79,80] in the functional point of view. However, more functions are expected from the scanner here. The scanner sequentially polls the status

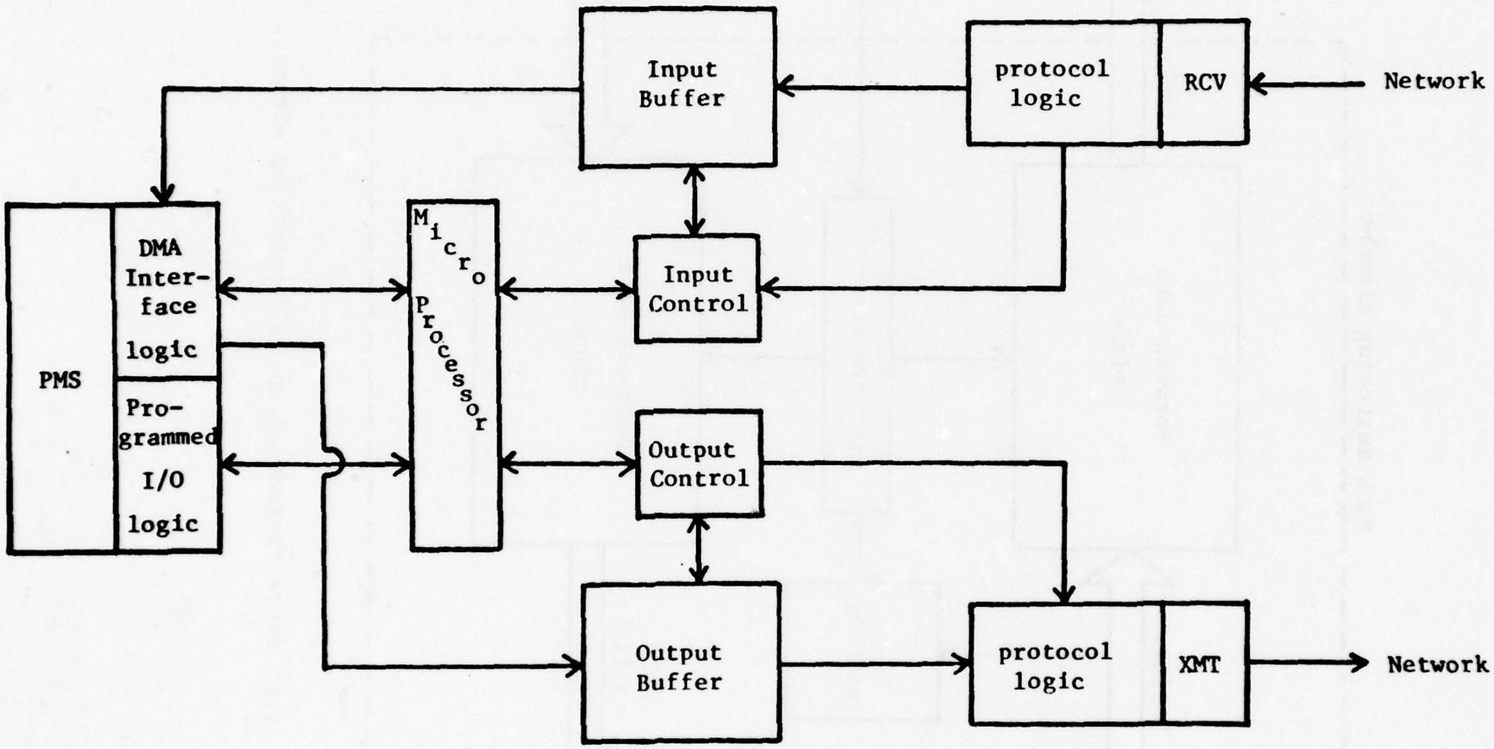


Fig. A.4 Functional block diagram of interface processor (IP).

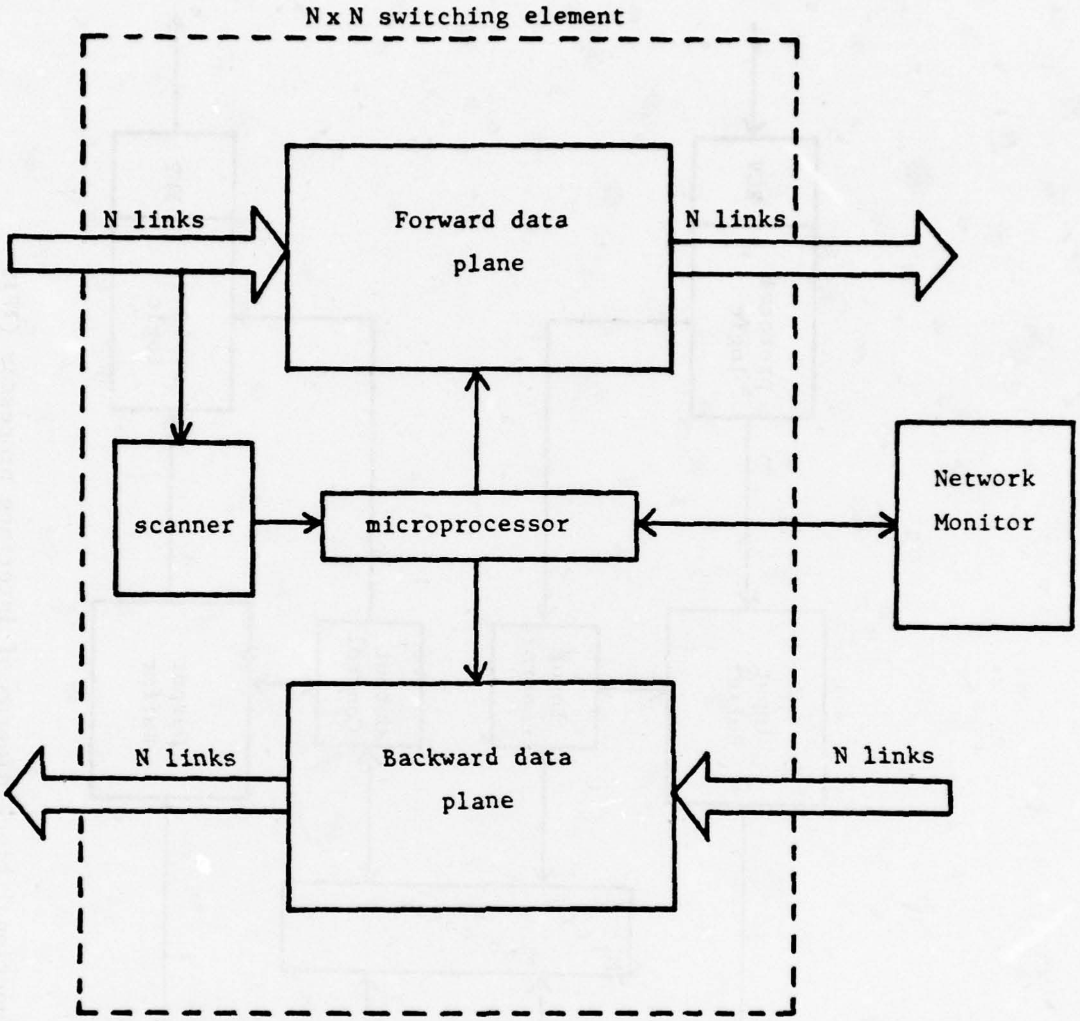


Fig. A.5 Block diagram of an N x N switching element.

of the input lines of the forward data plane. Upon detecting a connection in the protocol conversion logic of the scanner, the polling address and the extracted destination address are passed to the microprocessor. If an open signal is detected, the polling address is passed to the microprocessor for disconnecting the path.

The microprocessor receives the connection address and the disconnection address from the scanner and transforms these messages into control signals which are sent to the data planes for establishing the path or disconnecting the path. If there is a path conflict existing in the data plane for adding a new path, the control signals are then ignored. The routing procedure which is built in RAM not only generates the control signals but also monitors the routing statistics which are then sent to the network monitor for updating the routing table and for diagnostic purposes.

There are several N × N interconnection networks which can be used to implement the circuit switching required in the forward data plane or in the backward data plane. We will consider three kinds of interconnection networks: the versatile line manipulator [39], the class of multistage interconnection networks, and the Benes binary network [26]. An N × N versatile line manipulator (VLM) is shown in Fig. 2.19. The jth input is connected to the ith output if the cell (i,j) is activated. The ith address control register determines the location of the cell in the ith row to be activated. The VLM can also provide the capability of one-to-many connections. The configurations of the class of multistage interconnection networks have been described in Chapter 3. The Benes network can be obtained by overlapping the rightmost stage of the baseline network and the leftmost stage of the reverse baseline network as described in Chapter 5. In addition to the interconnection network body, a set of register array for controlling the 2 × 2 switching element and a decoder for distributing control signals to the proper register array, as shown in Fig. 2.19 for the versatile line manipulator, should be incorporated into each data plane.

C. Network monitor

The network monitor functions as a statistic analyzer and a diagnostic unit. It is connected to the network in the same way as the PMS' are connected. Hence it can receive messages from each PMS and also can transfer messages to each PMS. The network monitor can also directly access each N × N switching element to collect traffic statistics and update routing tables.

A.4 Software Control

This section discusses the communication protocols, routing techniques, reliability, and the overall operation of the network.

A. Communication protocols

The two-level communication protocols of circuit switching data networks recommended by several standard organizations such as ANSI and CCITT [78] have been utilized here. Level 1 concerns the physical and electrical characteristics to establish, maintain, and disconnect the physical link between devices and Level 2 describes the point-to-point link control for the exchange data between two devices. As shown in Fig. A.3, each block represents a device point in a path. The point-to-point communication occurs between the following points: from source PMS to source IP, from source IP to each N × N switching element, from source IP to destination IP and from destination IP to destination PMS. The communication protocols are implemented in the IP and the scanner hardware as described in Section A.3. The data and the supervisory information are packed into the hardware generated frame. The flow and error control are also included in Level 2 protocol.

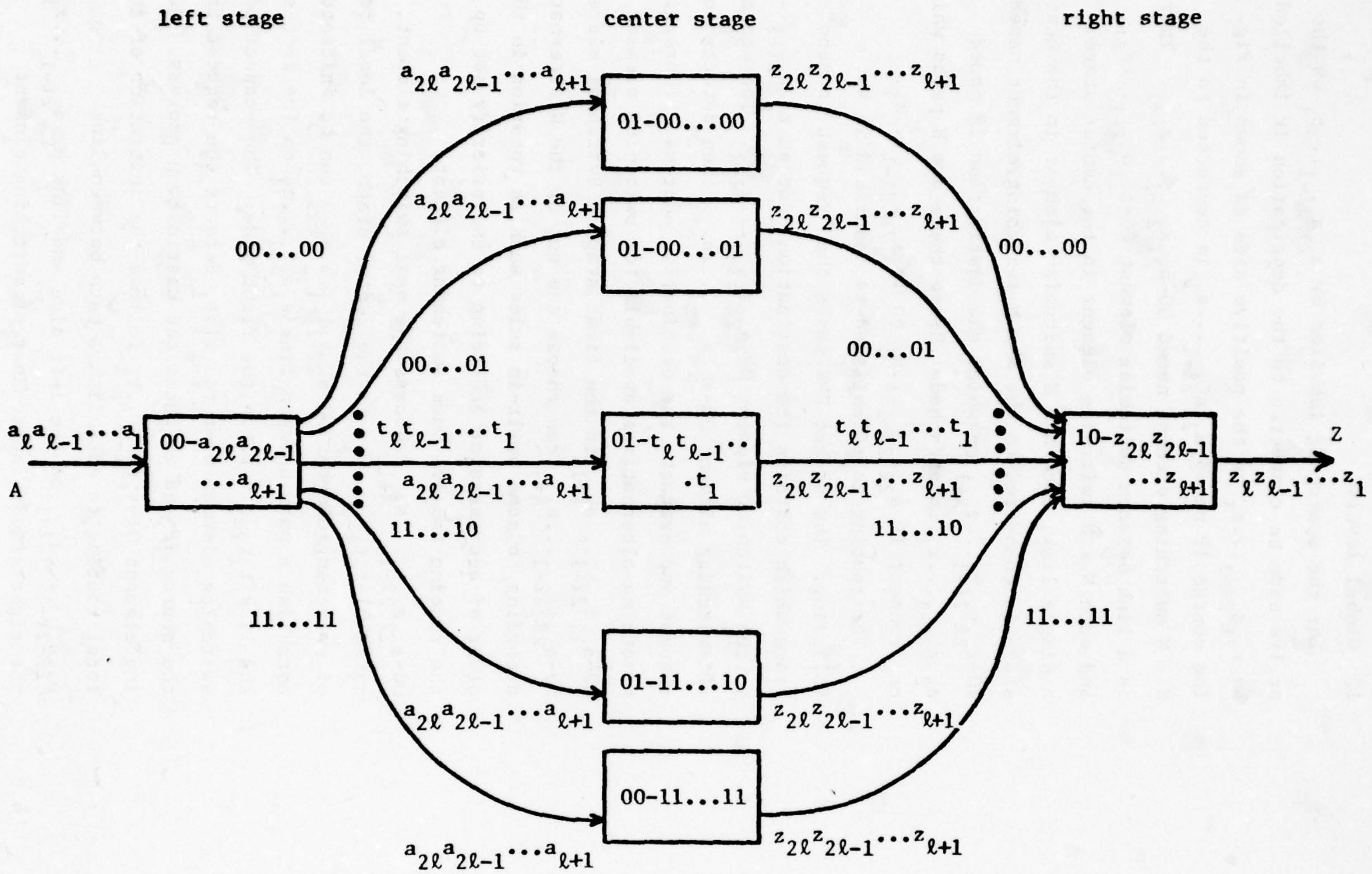
B. Routing techniques

The routing problem can be considered from two levels: global level and local level. The global level considers the interpoint routing problem and the local level concerns the routing problem inside the N × N switching element.

1. Global level:

Let the source IP labelled by $a_{2\ell}a_{2\ell-1}\cdots a_1$ on the active side be connected to the destination IP labelled by $z_{2\ell}z_{2\ell-1}\cdots z_1$ on the positive side as shown in Fig. A.6. The source IP named $a_{2\ell}a_{2\ell-1}\cdots a_1$ is connected to the N × N switching element named 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$. There is a link between switching element 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$ and each N × N switching element in the center stage. By a single link, each N × N switching element in the center stage is connected to the N × N switching element named 10-$z_{2\ell}z_{2\ell-1}\cdots z_{\ell+1}$ on which the destination IP named $z_{2\ell}z_{2\ell-1}\cdots z_1$ is attached. Hence there are N paths which can connect IP $a_{2\ell}a_{2\ell-1}\cdots a_1$ to IP $z_{2\ell}z_{2\ell-1}\cdots z_1$.

The routing is established as described in the following. The source PMS sends the "request to send" frame which contains the destination address $z_{2\ell}z_{2\ell-1}\cdots z_1$ to the switching element 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$. The scanner of switching element 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$ then detects the request and extracts the destination address. There are N routing alternatives available for switching element 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$ in the first stage. Switching element 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$ can choose one out of the N alternatives according to some built-in rules such as rotation in the order of addresses or according to the priority set up in the routing table. Once switching element 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$ chooses the next switching element, named 01-$t_{\ell}t_{\ell-1}\cdots t_1$, in the center stage, the local routing of switching element 00-$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$ can be initiated to establish a path between link $a_{\ell}a_{\ell-1}\cdots a_1$ on the left side and link $t_{\ell}t_{\ell-1}\cdots t_1$ on the right side. The scanner of switching element 01-$t_{\ell}t_{\ell-1}\cdots t_1$ detects the request from the source IP and extracts the destination address. Switching element 01-$t_{\ell}t_{\ell-1}\cdots t_1$ follows the initiation of the local routing to establish a path between link $a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$ on the left side and link $z_{2\ell}z_{2\ell-1}\cdots z_{\ell+1}$ on the right side. The scanner of switching element



left stage
center stage
right stage
$a_{\ell}a_{\ell-1}\cdots a_1$
A
$00\text{-}a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$
00...00
00...01
11...10
11...11
$a_{2\ell}a_{2\ell-1}\cdots a_{\ell+1}$
$t_{\ell}t_{\ell-1}\cdots t_1$
01-00...00
01-00...01
$01\text{-}t_{\ell}t_{\ell-1}\cdots t_1$
01-11...10
00-11...11
$z_{2\ell}z_{2\ell-1}\cdots z_{\ell+1}$
$10\text{-}z_{2\ell}z_{2\ell-1}\cdots z_{\ell+1}$
Z
$z_{\ell}z_{\ell-1}\cdots z_1$

Fig. A.6 Glocal routing (from A = $a_{2\ell}a_{2\ell-1}\cdots a_1$ to Z = $z_{2\ell}z_{2\ell-1}\cdots z_1$).

10-$z_{2\ell}z_{2\ell-1}\dots z_{\ell+1}$ in the right stage then detects the request from source IP and extracts the destination address. Switching element 10-$z_{2\ell}z_{2\ell-1}\dots z_{\ell+1}$ finally establishes a path between input $t_{\ell}t_{\ell-1}\dots t_{1}$ on the left side and output $z_{\ell}z_{\ell-1}\dots z_{1}$ on the right side. This completes a bidirectional path between source PMS' IP and destination PMS' IP. Since there are alternative paths for a connection request, an optimal routing problem naturally arises. An adaptive routing method can be used by implementing a dynamic routing table which is optimally updated by the network monitor.

2. Local level:

The local level routing of three kinds of interconnection networks as described in Section A.3.B will be investigated here.

a. Versatile line manipulator: The versatile line manipulator requires little logical complexity for the local routing. To connect a path, the controller of the N × N switching element checks into its working file for the availability of the addressed link on the destination side. If the link on the destination side is free, the controller sends the address of the two addressed terminal links, j and i, to the forward and backward data planes to update the IMR and OMR and to activate cell (i,j). To remove a path, the controller restores the mask bit in IMR and OMR, and the availability of the link on the destination side in its working file.

b. A class of multistage interconnection networks: A homogeneous routing procedure, names binary tree coding method, has been developed for the class of multistage interconnection networks in Chapter 3. How to control the 2 × 2 switching element for the asynchronous operation has been discussed in Chapter 6. See Fig. 6.5 for the grouping of the valid states and Eqs. (6.14) and (6.15) for the setting of the control bit. A status table of switching elements is constructed for the routing procedure. An example is shown in Table A.1. The routing problem is defined as follows. Suppose we have existing connections in the N × N switching elements and the status table is given as the example shown in part (a) of Table A.1, we want to add a new connection or to remove an existing connection. Part (a) of Table A.1 is used to show the status of an 8 × 8 baseline network which contains the connection between terminal 0 on Side 1 and terminal 4 on Side 2. Each column of the table corresponds to a stage. The upper row of the entry in the table is the status information of the 2 × 2 switching element specified in the lower row. The leftmost bit of the status information is the control bit and the other two bits are used to specify the number of users. Assume terminal 2 on Side 1 requests connection to terminal 7 on Side 2. According to Eq. (6.14) the in-path switching elements are $(01)_0$, $(10)_1$ and $(11)_2$. The related control bits are 0, 0 and 1 according to Eq. (6.15). First we check if conflicts exist between the existing connections and the one to be added by applying a bit-by-bit EXCLUSIVE OR operation between the status information of each in-path switching element and the code word formed by concatenating the calculated control bit and 00. If no result of the operation is greater than 100, there is no conflict and we can proceed to update the status table. Otherwise, the request should be deferred. The control signal should be sent to the switching element if the result of the EXCLUSIVE OR operation for that switching element is equal to 100. There is no conflict in our example so that we proceed to update the status information. The contents of entries $(01)_0$ and $(11)_2$ should be changed from 000 to 001 and 101, respectively, and the contents of entry $(10)_1$, should be changed from 001 to 010 since there are two users using $(10)_1$. The updated table is shown in part (b) of Table I.

A control signal 1 should be sent to $(11)_2$. The removal of a connection can be done by decreasing the number shown in the user field of the status information by 1.

c. Benes network: Some routing algorithms [29,73,74] have been developed to perform permutations on the Benes network. These routing algorithms are not suitable for the asynchronous operation. A new routing algorithm for the asynchronous operation is developed here. Using the routing procedure developed for the class of multistage interconnection networks in Chapter 3, we obtain the following properties:

(1) There are N/2 possible paths connecting two terminal links at the opposite sides of the N × N Benes network.

(2) Each of the N/2 possible paths passes through different 2 × 2 switching element in the center state.

(3) Let $A = a_\ell a_{\ell-1} \dots a_1$ on the left side be connected to $Z = z_\ell z_{\ell-1} \dots z_1$ on the right side and the two valid states described for the class of multistage interconnection networks above also be used for the Benes network. The control bit of the in-path switching element in the center stage is equal to $a_\ell \oplus z_\ell$.

An example of an 8 × 8 Benes network is shown in Fig. A.7. The source terminal A = 011 is connected to destination terminal Z = 101 via N/2 (N=8) paths. Each path passes through different 2 x 2 switching element in the center stage and the control bit of each switching element should be equal to $a_2 \oplus z_2 = 1$. According to the above properties and the status table developed as shown in Table A.1, we have a simple routing algorithm for the asynchronous operation:

Step 1: Calculate the control bit of the in-path switching element in the center stage using $a_\ell \oplus z_\ell$.

Step 2: Find a free 2 × 2 switching element in the center

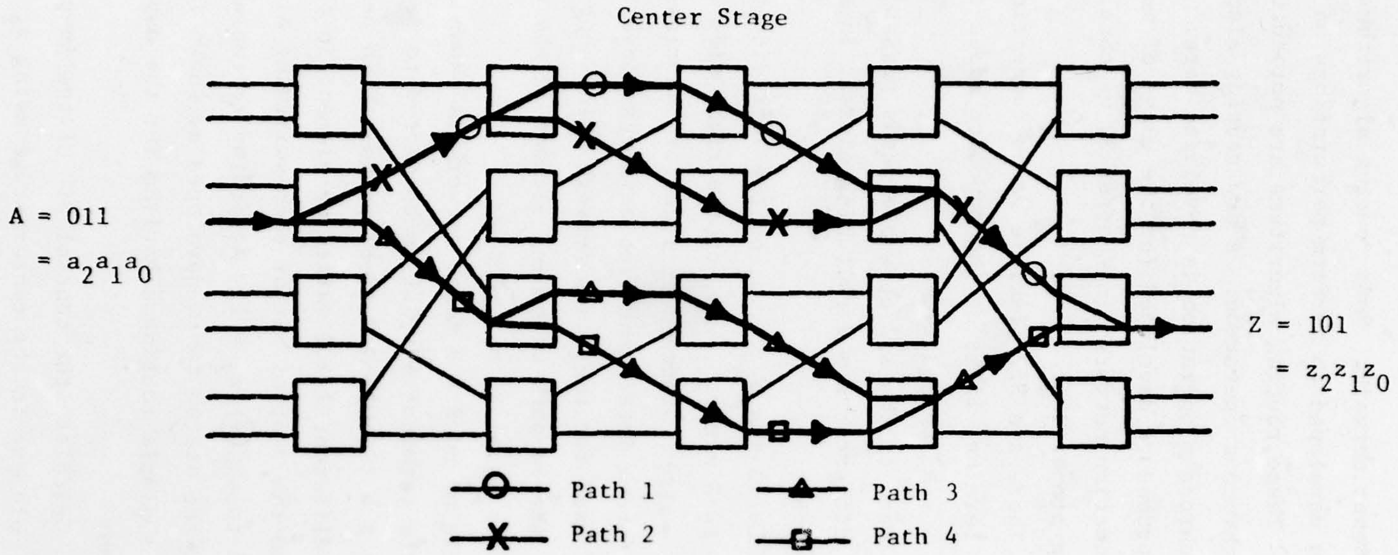


Fig. A.7 Example of N/2 paths

Table A.1 Status Table of an 8 x 8 Baseline Network for asynchronous Operation.

(a) Status Table.

(0 → 4)

101 $(00)_0$	000 $(00)_1$	000 $(00)_2$
000 $(01)_0$	000 $(01)_1$	000 $(01)_2$
000 $(10)_0$	001 $(10)_1$	001 $(10)_2$
000 $(11)_0$	000 $(11)_1$	000 $(11)_2$

(b) Status Table.

(0 → 4; 3 → 7)

101	000	000
001	000	000
000	010	010
000	000	101

stage which can be set by the calculated control bit.

Step 3: Assume the location of the in-path links connected to the center stage are L and R in the left and right sides of the 2 × 2 switching element in the center stage, respectively. Calculate the routing information using A and R as the source-destination pair on the baseline network and using L and Z as the source-destination pair on the reverse baseline network.

Step 4: Do the routing procedure as described in the example shown in Table A.1. If there are conflicts go to Step 2 to try another path.

C. Reliability

The circuit switching network can provide multiple paths between any two PMS' while connections from various switching elements to multiports of PMS enables one of the several alternate paths to be established. The failure of one N × N switching element does not rule out the connection possibility between any two PMS' and does not affect the function of other N × N switching elements. A failure can be reported to the network monitor by PMS' via connection paths or by the controllers of N × N switching elements. A failure can be detected by PMS' if a message has been retransmitted several times without a response or with a negative acknowledge from the destination. An invalid sequence can also be detected by an N × N switching element as a failure. The use of Cycle Redundancy Code (CRC) allows continuous checking of the data and improves the reliability of the system. The fault diagnosis can then be initiated by the network monitor. A redundant standby network monitor can be used to take over the monitoring work whenever the operating one fails.

D. Overview of data transmission

All transmissions are in frames which sequentially contain initial flag sequence, destination address field, control field, information field, frame check bit sequence, and end flag sequence. Frame types include request to send (RS), ready to receive (RR), not ready to receive (NRR), acknowledge (ACK), negative acknowledge (NACK), and data.

The PMS selects a port and puts outgoing data in the output buffer of the IP via the DMA logic. When the data is ready for transmission, the IP hardware generates an RS frame and transmits it repeatedly to the network within a prescribed time interval. If the sending (or source) IP does not receive a response from the receiving (or destination) IP in the time interval, it retries after a delay period.

The scanner of the N × N switching element connected to the IP recognizes the RS and sends the source and the destination addresses to the controller. The controller then selects one N × N switching element in the center stage for connection, according to the routing table, and generates the control signals to connect the IP to that N × N switching element in the center stage on the forward data plane and the backward data plane. The scanner of the chosen N × N switching element in the center stage can now receive the RS frame. The destination address is extracted and the controller checks the availability of the link connected to the N × N switching element (in the right stage) which is connected to the destination. If the link is occupied the address extracted is just ignored and the link availability is rechecked in the next scanner cycle. If the link is not occupied the controller generates the control signals to connect the path from the N × N switching element in the left stage to the N × N switching element in the right stage. The scanner of the N × N switching element in the right stage can then receive the RS frame and the controller of that switching element can complete the path by generating and sending control signals to the data planes to connect the N × N switching element in the center stage to the receiving IP.

The receiving IP is interrupted after receiving the first RS and it verifies that it is the destination. If not, the RS frame is ignored and a fault is reported. If the receiving IP verifies that the destination is itself, it sends an RR or NRR frame (not ready to receive). If the NRR is returned the sending IP terminates the transmission and retries after a prescribed time interval. If the RR is returned the sending IP is interrupted and it initiates transmission of the data frames. The receiving IP acknowledges a valid reception with an ACK frame and then initiates a DMA request to transfer the data in the input buffer to the PMS, or acknowledges an error reception with NACK frame. If an ACK frame is returned the sending IP is interrupted and sends an open signal. The scanner of each N × N switching element in the path recognizes the open signal and the controllers in the path accordingly disconnect the path. If an NACK is returned, the retransmission is initiated. If the receiving IP responds to the NACK frame again, the sending IP then sends open signals to break the path. After initiating a report of the presence of the retransmission fault the PMS can then select another port and initiate the same connection procedure to transmit the data.

A.5 Discussion of System Characteristics

The network contains 3N N × N switching elements and hence requires 3N microcomputers for the distributed routing control. The functionally and physically equivalent N × N switching elements are good for the cost effective LSI implementation and the software development. Since the number of ports is N^2 on the right side and N^2 on the left side, $2N^2$ additional microcomputers are required for the interface processors which implement the communication protocols at the interface of the PMS and the network.

Independent multiple paths between any two PMS' facilitate the graceful degradation capability. The failure of an N × N switching element does not affect the operation of other N × N switching elements. The dual network monitors serve the diagnostic functions.

In contrast to the flood routing procedure, the routing procedure developed here allows simultaneous path establishments for connection requests. The adaptive routing scheme can be implemented to achieve the optimal routing.

The network can simultaneously provide N^2 full duplex, asynchronous, bit serial circuit switching paths. The bit serial path can be extended to the bit parallel path by adding the proper number of data planes in each N × N switching element and using bit parallel IPs. The throughput of each path is dependent on the transmission speed of the interface microcomputer using DMA techniques. The Fairchild F464 CCD memory can provide a transfer rate of 5 Mbit/sec. Hence the maximum network throughput can be as high as $5N^2$ Mbit/sec. It should be remembered that significantly higher speed memory and devices can be expected in the future.

The blocking probability of establishing a path and the average response time greatly depends on the size and amount of traffic, the routing strategy and the hardware structure of the data planes. The traffic model is different from case to case. An adaptive routing method can be utilized to achieve optimal routing. We have illustrated three different hardware alternatives for the data planes: the versatile line manipulator, the class of multistage interconnection networks and the Benes network. Among the three networks the versatile line manipulator requires the least amount of complex software with the maximum amount of logic circuits (about $N/\log_2 N$ times that of the class of multistage interconnection networks). The class of multistage interconnection networks provides the highest blocking probability and requires the simplest hardware configuration while almost doubly complex hardware and software is needed for the Benes network. However, the Benes network can provide N/2 alternative paths for the establishment of a connection. A detailed simulation study is required to observe the response time characteristics under different traffic models, routing strategies and data plane implementation.

REFERENCES

[1] T. Feng (Editor), Special Issue: Parallel Processors and Processing, ACM Computing Surveys, Vol. 9, No. 1, March 1977.

[2] M. J. Flynn, "Very high-speed computing systems," Proc. of the IEEE, Vol. 54, No. 12, Dec. 1966, pp. 1901-1909.

[3] A. W. Burks, H. H. Goldstine, and J. von Neumann, "Preliminary dicussion of the logical design of an electronic computing instrument ," Part I, Datamation, Vol. 8, No. 9, Sept. 1962, pp. 24-31. Part II , Datamation, Vol. 8, No. 10, Oct. 1962, pp. 36-41.

[4] T.Feng, "Some characteristics of associative parallel processing," Proc. of the 1972 Sagamore Computer Conference, pp. 5-16.

[5] P. H. Enslow, Jr., Multiprocessors and Parallel Processing, John Wiley and Sons, New York, 1974.

[6] J. L. Baer, "Multiprocessing system," IEEE Trans. Comput, Vol. C-25, No. 12, Dec. 1976, pp. 1271-1277.

[7] M. Abrams, R. P. Blanc and I. W. Cotton, (Editors), Computer Networks : Text and reference for a tutorial, JH3100-5C, IEEE Inc., 1976 Revision.

[8] P. H. Enslow, Jr., "What is a distributed data processing system?," Computer, Vol. 11, No. 1, Jan. 1978, pp. 13-21.

[9] K. J. Thurber and G. M. Masson, "Recent advances in microprocessor technology and their impact on interconnection design in computer systems," 1977 IEEE International Conference on Communication Record, Vol. 3, 1977, pp. 46.2/216-46.2/220.

[10] S. I. Kartashev and S. P. Kartashev (Editors), Special Issue: Modular Computers and Networks, Computer, Vol. 11, No. 7, July 1978.

[11] G. J. Lipovski and K. L. Doty, "Developments and directions in computer architecture," Computer, Vol. 11, No. 8, Augst 1978, pp. 54-67.

[12] D. P. Siewiorek, D. E. Thomas and D. L. Scharfetter, "Use of LSI modules in computer structures: Trends and limitations," Computer, Vol. 11, No. 7, July 1978, pp. 16-25.

[13] P. H. Enslow, Jr., "Multiprocessor organization - A Survey," Proc. of the 1975 Sagamore Computer Conference on Parallel Processing, pp. 63-70.

[14] E. A. Anderson and E. D. Jensen, "Computer interconnection structures : Taxanomy, characteristics, and examples," ACM Computing Survey, Vol. 7, No. 4, Dec. 1975, pp. 197-213.

[15] K. J. Thurber, et al., " A systematic approach to the design of digital bussing structures," Proc. AFIPS 1972 FJCC, pp. 719-740.

[16] GTE Sylvania Inc., Supplemental Conceptual Design Study of an Integrated Voice/Data Switching and Multiplexing Technique for an Access Area Exchange, report submitted to the Defense Communications Agency, November 11, 1976.

[17] M. Barbacci, et al., The Application of Multiple Processor Computer Systems to Digital Communication Networks, Carnegie Mellon University report, June 22, 1976.

[18] W. A. Wulf and C. C. Bell, "C.mmp: A multi-mini-processor," Proc. AFIPS 1972 FJCC, pp. 765-777.

[19] K. E. Batcher, "The flip network in STARAN," Proc. of the 1976 International Conference on Parallel Processing, pp. 65-71.

[20] F. E. Heart, S. M. Ornstein, W. R. Crowther, and W. B. Barker, "A new mini-computer/multiprocessor for the ARPA network," Proc. AFIPS 1973 National Computer Conference, pp. 529-537.

[21] S. M. Ornstein, W. R. Crowther, M. F. Kraley, R. D. Bressler, A. Michel, and F. E. Heart, "Pluribus - A reliable multiptocessor," Proc. AFIPS 1975 National Computer Conference, pp. 551-559.

[22] C. Clos, "A study of nonblocking switching network," Bell Syst. Tech. J., Vol. 32, 1953, pp. 406-424.

[23] D. Cantor, "On contruction of nonblocking switching networks," in Proc. Symp. Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, Brooklyn, NY, 1972, pp. 253-255.

[24] D. Cantor, "On nonblocking switching networks," Networks, Vol. I, 1972, pp. 367-378.

[25] M. J. Marcus, "The theory of connecting networks and their complexity : a review," Proc. of the IEEE, Vol. 65, No. 9, Sept. 1977, pp. 1263-1270.

[26] V. Benes, Mathematical Theory of Connecting Networks, New York: Academic Press, 1965.

[27] A. Waksman, "A permutation network," J. Association Comput. Machine, Vol. 15, 1968, pp. 159-163.

[28] A. Joel, "On permutation networks," Bell Syst. Tech. J., Vol. 67, 1968, pp. 813-822.

[29] D. Opferman and N. Tsao-Wu, "On a class of rearrangeable switching networks," Bell Syst. Tech. J., Vol. 50, May-June 1971, pp. 1579-1618.

[30] K. J. Thurber, "Interconnection network - A survey and assessment," Proc. AFIPS 1974 National Computer Conference, pp. 909-919.

[31] N. Pippenger, "On crossbar switching networks," IEEE Trans. Commun., Vol. COM-23, June 1975, pp. 646-659.

[32] G. J. Lipovski, "The architecture of a large associative processor," Proc. AFIPS 1970 SJCC, pp. 385-396.

[33] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessing systems," Proc. 1st Annual Computer Architecture Conf., Dec. 1973, pp. 21-28.

[34] G. J. Lipovski and A. Tripathi, " A reconfigurable varistructure array processor," Proc. of the 1977 International Conference on Parallel Processing, PP. 165-174.

[35] D. H. Lawrie, Memory-Processor Connection Networks, University of Illinois Report UIUCDCS-R-73-557, Feb. 1973.

[36] D. H. Lawrie, "Access and alignment of data in an array processor," IEEE Trans. Comput.,Vol. C-24, Dec. 1975, pp. 1145-1155.

[37] T. Feng, Parallel Processing Characteristics and Implementation of Data Manipulating Functions, Technical Report, RADC-TR-73-189, July 1973, 766279/4.

[38] T. Feng, "Data manipulating functions in parallel processors and their implementations," IEEE Trans. Comput.,Vol. C-23, No. 3, March 1974, pp. 309-318.

[39] T. Feng, The Design of a Versatile Line Manipulator, Tech. Report RADC-TR-73-292, Sept. 1973, 773172/2GI.

[40] W. W. Gaertner, M. P. Patel, C. T. Retter, and I. M. Singh, "Construction of a versatile data manipulator for parallel/associative processors," Proc. of the 1976 International Conference on Parallel

Processing, p. 72.

[41] M. C. Pease, "An adaptation of the fast Fourier transform for parallel processing," J. Association Comput. Machine, Vol. 15, April 1968, pp. 252-264.

[42] H. S. Stone, "Parallel processing and the perfect shffle," IEEE Trans. Comput.,Vol. C-20, No. 4, April 1972, pp. 357-366.

[43] T. Lang and H. S. Stone, "A shuffle-exchange network with simplified control," IEEE Trans. Comput.,Vol. C-25, No. 1, Jan. 1976, pp. 55-65.

[44] K. E. Batcher, "The multi-dimensional-access memory in STARAN," Proc. of the 1975 Sagamore Computer Conference, p. 167; also in IEEE Trans. Comput., Vol. C-26, No. 2, Feb. 1977, pp. 174-177.

[45] M. C. Pease, "The indirect binary n-cube microprocessor array," IEEE Trans. Comput., Vol. C-26, No.5, May 1977, pp. 548-573.

[46] W. H. Kautz, et al., "Cellular interconnection arrays," IEEE Trans. Comput., Vol. C-17, No. 5, May 1968, pp. 443-451.

[47] J. Gecsei, "Interconnection networks from three state cells," IEEE Trans. Comput., Vol. C-26, No. 8, Aug. 1977, pp. 705-711.

[48] H. J. Siegel, "Analysis techniques for SIMD machine interconnection networks and the effects of processor address masks," IEEE Trans. Comput., Vol. C-26, No. 2, Feb. 1977, pp.153-161.

[49] H. J. Siegel , "Single instruction stream-multiple data stream machine interconnection network design," Proc. of the 1976 International Conference on Parallel Processing, pp. 272-282.

[50] H. J. Siegel and S. D. Smith, "Study of multistage SIMD interconnection network," Proc. Fifth Annual Symp. on Comput. Architecture , April 1978, pp. 223-229.

[51] J. B. Dennis, "Packet communication architecture," Proc. of the 1975 Sagamore Computer Conference, pp. 224-229.

[52] H. Sullivan, T. R. Bashkow, and K. Klappholz, "a large scale homogeneous, fully distributed parallel machine," Proc. 4th Annual Symp. on Computer Architecture, Nov. 1977, pp. 105-125.

[53] E. A. Harrington, "Synchronization techniques for various switching network toptlogies," IEEE Trans. Commun., Vol. COM-26, No. 6, June

1978, pp. 925-932.

[54] J. G. Fletcher, "Serial communication protocol simplified data transmission and verification," Computer Design, July 1978, pp. 77-86.

[55] A. Kershenbaum, "Tools for planning and designing data communcation network," Proc. AFIPS 1974 National Computer Conference, pp. 583-591.

[56] D. C. Stanzione, "Microprocessor in telecommunication systems," Proc. of the IEEE, Vol. 66, No. 2, Feb. 1978, pp. 192-199.

[57] R. C. Chen, P. G. Jessel and R. A. Patterson, "Mininet: A microprocessor-controlled mininetwork," Proc. of the IEEE, Vol. 64, No. 6, June 1976, pp. 988-993.

[58] A. Dhawan and D. Mueller, "An application of a microprocessor in a large circuit/packet switching system," IEEE International Conference on Communication Record, Vol. 3, 1977, pp. 47.3/237 - 47.3/241.

[59] W. Keister, R. W. Ketchledge and H. E. Vaughan, "No. 1 ESS: System organization and objectives," Bell Syst. Tech. J., Vol. 43, Sept. 1964, pp. 1831-1844.

[60] O. Holger and L. Kleinrock, "The influence of control procedures on the performance of packet-switched networks," National Telecommun. Conference 1974 Record, pp. 810-817.

[61] C. A. Dahlbom, "Signaling system and technology," Proc. of the IEEE, Vol. 65, No. 9, Sept. 1977, pp. 1349-1353.

[62] R. E. Kahn and W. R. Crowther, "Flow control in a resource-sharing computer network," Proc. ACM/IEEE 2nd Symp. on Problems in Optimization of Data Commun. Sept., Oct. 20-22, 1971, pp. 108-116.

[63] D. W. Daves, "The control of congestion in packet switching networks," Proc. ACM/IEEE 2nd Symp. on Problems in Optimization of Data Commun. Sept., Oct. 20-22, 1971, pp. 46-49.

[64] H. Frank, "Providing reliable network with unreliable components," Proc. 3rd IEEE Data Commun. Symp., Nov. 1973, pp. 161-164.

[65] W. Crowther, J. McQuillan and D. Walden, "Reliablity Issues in the ARPA network," Proc. 3rd Data Commun. Symp., Nov. 1973, pp. 159-160.

[66] R. L. Graham and H. O. Pollak, "On the addressing problem for loop

switching," Bell Syst. Tech. J., Vol. 50, Oct. 1971, pp. 2495-2519.

[67] J. M. McQuillan, "Routing algorithm for computer networks - A Survey," National Telecommun. Coference 1977 Record, pp. 28:1/1 - 28:1/6.

[68] J. M. McQuillan, "Design considerations for routing algorithm in computer networks," Proc. 7th Hawaii Int'l. Conf. Syst. Sci., Jan. 1974, pp. 22-24.

[69] K. E. Batcher, "Sorting networks and their applications," Proc. AFIPS 1968 SJCC, pp. 307-314.

[70] S. W. Golomb, "Permutations by cutting and shuffling," SIAM REV., Vol. 3, Oct. 1961, pp. 293-297.

[71] H. S. Stone, "Dynamic memories with enhanced data access," IEEE Trans. Comput., Vol. C-21, No. 4, April 1972, pp. 359-366.

[72] T. Lang, "Interconnections between processors and memory modules using the shuffle-exchange network," IEEE Trans. Comput., Vol. C-25, No. 5, May 1976, pp. 496-503.

[73] S. Andersen, "The looping algorithm extended to base 2^t rearrangeable switching networks," IEEE Trans. Commun., Vol. COM-25, No. 10, Oct. 1977, pp. 1057-1063.

[74] J. Lenfant, "Parallel permutations of data: A Benes network control algorithm for frequently used permutations," IEEE Trans Comput., Vol. C-27, No. 7, July 1978, pp. 637-647.

[75] R. C. Swanson,"Interconnections for parrllel memories to unscramble p-ordered vectors," IEEE Trans . Comput., Vol. C-23, No. 11, Nov. 1974, pp. 1105-1116.

[76] North Electric Company, Communication Processor Systems, Tech. Report RADC-TR-76-394, Vol. VIII, Jan. 1977, A036873.

[77] J. F. Springer,"The distributed data network,its architecture and operation," Proc. COMPCOM FALL 1978, pp. 221-228.

[78] E. A. Shearin and L. L. King,"Microprocessor implementation of CCITT recommendation X.25 (Level 1 & 2) protocal," Proc. of Computer Networking Symposium, Dec. 1977, pp. 125-130.

[79] L. Freimanis, A. M. Guercio, and H. F. May,"No. 1 ESS scanner, signal distributor and central pulse distributor," Bell Syst. Tech. J., Vol. 43, Sept. 1964, pp. 2254-2282.

[80] A. K. Shrivastava et al.,"Autonomous line scanning for SPC telephone switching system," IEEE Trans. Commun., Vol. COM-26,No. 3, March 1978, pp. 368-373.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.